# Distributed Intelligence for Air Fleet Control

Randall Steeb, Stephanie Cammarata,
Frederick A. Hayes-Roth, Perry W. Thorndyke,
Robert B. Wesson

Rand

The research described in this report was sponsored by the Defense Advanced Research Projects Agency under ARPA Order No. 3640/3473, Contract No. MDA903-78-C-0029, Information Processing Techniques.

Library of Congress Cataloging in Publication Data
Main entry under title:

Distributed intelligence for air fleet control.

"R-2728-ARPA."
Bibliography: p.
1. Air traffic control. I. Steeb, Randall, 1946- . II. United States. Defense Advanced Research Projects Agency. III. Rand Corporation.
TL725.3.T7D57 629.136'6 81-13913
ISBN 0-8330-0353-4 AACR2



Published by The Rand Corporation

R-2728-ARPA

# Distributed Intelligence for Air Fleet Control

Randall Steeb, Stephanie Cammarata,
Frederick A. Hayes-Roth, Perry W. Thorndyke,
Robert B. Wesson

October 1981

Prepared for the
Defense Advanced Research Projects Agency

Rand
SANTA MONICA, CA. 90406

# PREFACE

This report summarizes the results of a nine-month investigation of distributed intelligence for air fleet control, conducted for the Information Processing Techniques Office, Defense Advanced Research Projects Agency, under Contract No. MDA 903-78-C-0029. The work has focused on development of organizations for cooperative problem-solving in complex, spatially distributed systems, using air traffic control as an illustrative context. The results should interest practitioners and researchers involved in developing systems or methods for distribution of tasks among a team of spatially or functionally distinct processors. The research follows some of the directions and approaches established in previous Rand research projects on situation assessment and planning. Related research is reported in the following Rand publications:

*Network Structures for Distributed Situation Assessment*, by R. Wesson and F. Hayes-Roth, with J. Burge, C. Stasz, and C. Sunshine, R-2560-ARPA, August 1980.

*Cognitive Processes in Planning*, by Barbara Hayes-Roth and Frederick Hayes-Roth, R-2366-ONR, December 1978.

*Dynamic Planning: Searching Through Time and Space*, by R. Wesson and F. Hayes-Roth, P-6266, February 1979.

# SUMMARY

This report examines several means of using distributed artificial intelligence—heuristic methods whereby multiple processors cooperate to achieve a common set of objectives—for commercial air traffic control (ATC). The ATC task consists of the ground- or air-based coordination of multiple aircraft with different performance parameters and destinations. The controller is responsible for maintaining separation between vehicles, environmental data-gathering, and route planning for airport arrivals, departures, and overflights. These activities demand complex, time-stressed, communications-rich, and spatially distributed problem-solving. Distributed planning and control techniques may be more effective than conventional centralized control in such environments because they offer greater speed, reliability, and flexibility, and because they minimize the need for costly long-distance communications.

This report focuses on the development of a number of system architectures for distributed situation assessment, planning, and control. The design kernel of these architectures is a model of the processing activities required for successful control of aircraft. The model comprises processing "experts" that share a common data base, or world model. These experts either sense the world and infer aircraft intentions, generate plans, evaluate plans, control and monitor plan execution, or communicate with other processors.

These functions may be distributed among multiple processors in a variety of ways. We postulate six distinct architectures for such distribution:

- Space-centered (based on geography).
- Function-centered (based on aircraft function).
- Plan-centered (based on planning approach).
- Hierarchical (based on level of abstraction).
- Object-centered autonomous (a silent, autonomous structure based on aircraft self-planning).
- Object-centered cooperative (a self-planning-based structure in which communication is used).

To illustrate the knowledge required and the methods employed in such architectural implementations, we outline a system design for the object-centered cooperative structure.

The utility of these architectures in ATC or other distributed problem-solving domains depends on environmental conditions, task characteristics and constraints, and available hardware. At least four factors are critical determinants of the optimal distributed architecture for a given domain: (1) degree of communication freedom, (2) extent of vehicle clustering, (3) time stress, and (4) reliability requirements. The first of these, communication freedom, seems to be the dominating factor. Communications bandwidth, noise, and range constrain the feasible designs for situation assessment, conflict recognition, planning, and bargaining methods.

# ACKNOWLEDGMENTS

This report benefited from the important contributions of several members of the Rand staff. James Gillogly adapted an ATC simulation for our experimental work. Keith Wescourt participated with the project group in informal experiments and provided many helpful suggestions. Gary Martins provided advice and management assistance throughout the project.

The ongoing research of several of our colleagues significantly enhanced our understanding of the problems faced in distributed planning and control. This research includes studies of situation assessment and planning by Frederick Hayes-Roth and Barbara Hayes-Roth, studies of air engagement simulation by Monti Callero and Philip Klahr, and the development of the ROSIE programming language by Daniel Gorlin, Frederick Hayes-Roth, and Henry Sowizral.

# CONTENTS

# FIGURES



# TABLES

# I. INTRODUCTION

Distributed problem-solving is an increasingly important process by which multiple problem-solvers cooperate to achieve their common objectives. Distributed computer systems using artificial intelligence (AI) programming methods have been investigated for a variety of domains, including battlefield intelligence-gathering (Cohen et al., 1979; Wesson et al., 1980), automated planning in enroute air traffic control (Andrews and Hollister, 1980), and automotive traffic light control (Brooks and Lesser, 1979). These domains are characterized by widely dispersed data-gathering, communications limitations, time-stressed decisionmaking, and natural clustering of activities. Distributed problem-solving uses separate processors to attack the problem at multiple points, exploiting parallelism for speed and power. Distribution also frequently entails the decomposition of the overall problem into a set of loosely coupled subproblems. This allows local data aggregation and planning and reduces the costly long-distance communications required by centralized systems.

In this report, we propose six distinct organizations of multiple processors for distributed problem-solving. We refer to these organizations as architectures because they serve as design frameworks for building large-scale systems. Each architecture provides a complete structure for sensing situational conditions, recognizing problems, planning cooperatively, and coordinating execution. The architectures differ in the way they decompose an overall problem, the extent and content of communications required between processors, and the type of cooperative behavior they produce.

We chose air traffic control (ATC) as the principal application area for development, testing, and implementation of our concepts, because ATC provides a complex, dynamic, time-stressed, and communications-rich problem-solving environment.

Currently, humans in ground-based centers control civilian air traffic. The airspace is divided into geographic sectors, each the responsibility of a different human controller. The controller coordinates operations within his sector and accepts control of aircraft handed off from the adjoining sectors. He is often heavily loaded with demands of pilot interrogation, track monitoring, flight-plan checking, plan generation, and clearance delivery. In addition, his performance is subject to degradation due to hardware failures, broadcast channel congestion, and human error (Couluris, Tashker, and Penick, 1978; Kinney, Spahn, and Amato, 1977).

Many of these problems could be minimized through the use of distributed problem-solving techniques. Situational sensing, conflict recognition, planning, and replanning can be accomplished by onboard equipment, for example, thereby reducing air-to-ground communications and their attendant response delays. Distributed ATC structures may also have greater reliability because the loss of a sensor, communication link, or processor should not seriously degrade overall system performance. This is true whether the distributed planning and control package is used as the primary system or as a backup.

To apply our distributed AI concepts to a practical situation, we developed a real-time ATC simulation and initial design specifications for a cooperative prob-

lem-solver. We chose the most interesting and intrinsically cooperative of our candidate architectures for this demonstration effort. This structure, which we term the *object-centered cooperative* organization, assigns an onboard processor to each aircraft and provides means for these processors to communicate locally. We describe the operation of a processor embedded in this architecture in a detailed scenario.

This report is organized into six sections: Section II describes the ATC task by showing its similarity to important military fleet control problems and by demonstrating how an AI kernel planner can represent its problem-solving functions. Section III presents six architectures for distributed planning and control and discusses the applicability of each to different ATC environments. Section IV considers in greater detail the relationships between architecture and environment, enumerating the influences of eight task environment conditions on communications networking, problem representation, plan generation, and bargaining protocols. Section V describes in detail the behavior of a single problem-solver in the object-centered cooperative architecture. Section VI summarizes the work and discusses our plans for future research.

# II. THE TASK OF AIR TRAFFIC CONTROL

In this section we present a model of the ATC task. We chose this task to develop, test, and demonstrate our distributed AI concepts because ATC entails multiple objects engaged in complex situation assessment and planning. A particular portion of the airspace may contain commercial jets converging on or departing from an airport, general aviation aircraft engaging in business or training activities, and/or military aircraft engaged in training exercises. Successful traffic supervision in this environment requires the coordination of many activities: maintaining inter-aircraft clearances, directing aircraft to avoid terrain and adverse weather conditions, collecting and interpreting data about the environment, monitoring system failures, and navigating to destinations.

In the following discussion, we describe the ATC functions from two perspectives: First, we show the similarities between ATC and other tasks such as cruise-missile coordination which share a number of goals that are common to all air fleet control. Second, we present a design for a kernel planner that embodies the expertise needed to perform ATC problem-solving functions.

## ATC AND AIR FLEET CONTROL

In ATC, a controller must achieve four goals:

1. *Error-free routing.* The controller must plan safe, executable routes across an airspace. These routes must avoid terrain obstacles, adverse weather conditions, and the routes of other aircraft.
2. *Uncertainty reduction.* The controller must often gather data to track the dynamic environment. The main forms of uncertainty in this environment involve positions of different aircraft in the airspace and changing traffic and weather conditions.
3. *Separation assurance.* The controller must monitor the locations of aircraft and avoid inter-vehicle collisions. This requires maintenance of safe horizontal and vertical separation standards. Depending on the airspace sector, federal regulations dictate a minimum of 3- to 5-mile horizontal separation and 1000-foot vertical separation between all aircraft (Andrews and Hollister, 1980; Rucker, 1979).
4. *Resource conservation.* The controller must select efficient routes, schedule delays, and minimize aircraft fuel consumption. To do so, he must plan routes that contain a minimum of large altitude changes and course alterations.

A number of air fleet control problems—e.g., coordination of cruise missiles, air intercept fighters, and remotely piloted vehicles—share these same goals. The correspondence between ATC goals and those in coordinating cruise missile missions is shown in Table 1. Error-free routing in both domains, for example, requires avoidance of terrain obstacles and weather. Uncertainty reduction in military oper-

Table 1

GOALS FOR AIR FLEET CONTROL: SIMILARITIES BETWEEN CIVILIAN ATC AND MILITARY CRUISE MISSILE CONTROL

| | Measurable Attributes | |
|---|---|---|
| Goal | Civilian ATC | Cruise Missile Control |
| Error-free routing | Conflicts in planned routes | Time within detection radius and acquisition envelope |
| Uncertainty reduction | Accuracy of weather and traffic models | Accuracy of weather and defense models |
| Separation assurance | Inter-aircraft distances | Inter-vehicle distances |
| Resource conservation | Fuel consumed; arrival delays | Fuel consumed; vehicle attrition; weapons utilization |

ations is somewhat more complex than in civilian ATC, as it entails locating enemy defenses and monitoring the status of the fleet's own supporting forces. Separation assurance between aircraft is required both for civilian flight and for military formation-keeping. Resource conservation, an economic consideration in ATC, becomes a strategic problem of minimizing vehicle attrition and allocating weapon expenditures efficiently in the military domain. The major difference between the domains lies in the relative importance of the various goals. ATC emphasizes separation between aircraft, while cruise missile missions focus on intelligence-gathering and maximizing penetration. Nevertheless, the same types of coordinated planning activities apply in both domains. Thus, AI techniques used to improve ATC should also apply to other areas of air fleet control.

## PROBLEM-SOLVING PROCESSES IN ATC

Coordination of air traffic requires integration of a number of complex problem-solving processes:

1. *Situation assessment*—generating and updating environmental and situational knowledge using sensor or communicated data.
2. *Conflict recognition*—identifying potential conflicts or problems requiring resolution.
3. *Plan generation*—synthesizing plans, refining them through the selection of appropriate actions, and replanning when necessary.
4. *Plan evaluation*—estimating probable outcomes of plan execution, aggregating costs and benefits, and determining the best option.
5. *Message passing*—exchanging data and plans via communication.

6. *Bargaining*—cooperating to resolve conflicts in plans or differences in situation assessments.
7. *Plan execution*—executing and monitoring selected plans.

To model the organization of these various activities, we adopted the "cooperating experts" paradigm of Hearsay-II (Erman et al., 1980). Figure 1 illustrates this organization. Each box represents an independent processing module, or expert, with its own goals and procedures for accomplishing those goals. The arrows between the boxes indicate data or results transferred between modules.

The various experts are organized around a *world model*, shown in the center of the figure. The world model contains knowledge of the locations, plans, and characteristics of the various known aircraft in the airspace. It posts requests from the experts for processing resources, results produced by these experts, and messages received from other controllers.

All modules use knowledge in the world model to support their activities, and they alter the contents of the model as part of their actions. The *sensor* receives radar returns, detects changes in the environment, and updates the current locations of aircraft in the world model. The *plan generator* uses the current situation estimate and demands for action to produce tentative plans or revisions to existing plans. The *evaluator* uses tentative plans and the current situation estimate, predicts the situation after a hypothetical execution of each plan, posts the conflicts that would result from the execution of the plans, and selects the best available plan. The *communicator* selectively exchanges data and plans with the other aircraft in the airspace. The controller then performs the tactical execution of actions at the times or positions specified by the plan.

We shall now consider how this organization of experts can accomplish the seven activities listed above.

*Situation assessment* involves updating the world model as new information becomes available. The sensor processes radar returns, compares the information with that in the world model, and sends updates regarding changed aircraft locations and environmental conditions. The communicator can also update the world model by posting locations, plans, and goals received from other processors.

*Conflict recognition* occurs in the evaluator module. The evaluator simulates execution of the pending flight plan and uses current situation estimates to identify potential conflicts. The fast-time simulation necessary to identify these conflicts can be performed in several different ways. The simplest method relies on trajectory projection, the technique currently used in ATC conflict-detection programs, in which straight-line extrapolation of an aircraft's current position and heading is used to predict its future location. A more accurate method of conflict recognition, plan-based simulation, includes planned maneuvers in the trajectory projections. This procedure requires the receipt (via the communicator) of current flight plans from the interacting aircraft.

*Plan generation* involves the synthesis of flight plans from origin to destination. The processors must generate fuel-efficient, conflict-free plans. When processors must share the use of the same airspace, they must coordinate their plans with one another. Four types of planning can be used to achieve this coordination: preplanned protocols, iterative local planning, asynchronous cooperative planning, or simultaneous global planning.

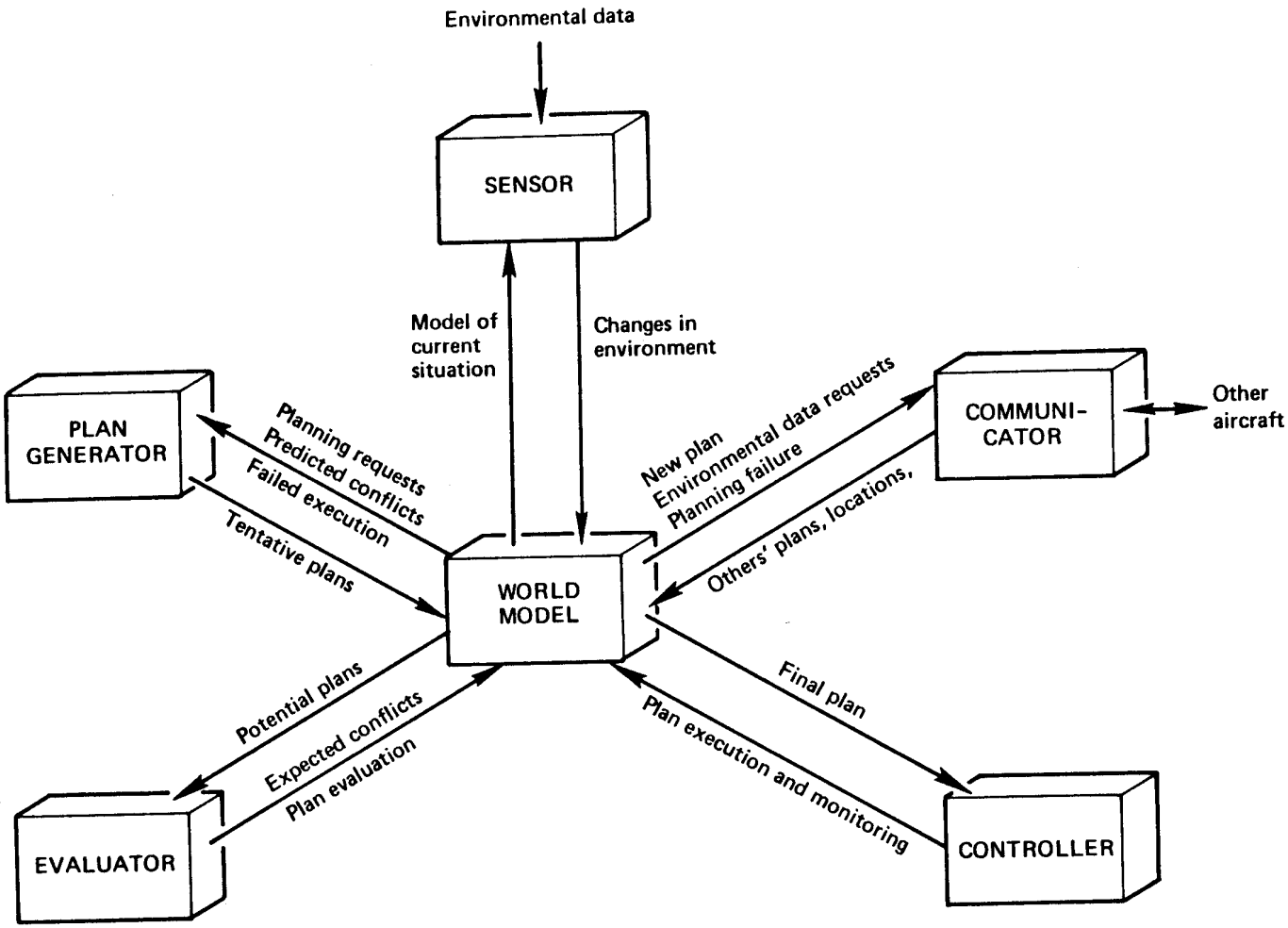


Fig. 1—The structure of the kernel planner

With preplanned protocols, each processor acts according to predefined rules known to all processors. Assuming that each processor maintains accurate knowledge of the planning methods and protocols of the other processors, communications are largely unnecessary. However, such rigid individual behavior may lead to suboptimal global solutions.

In iterative local planning, all processors sharing a local planning region initially exchange goals and constraints. For example, if two processors cooperate to resolve a particular conflict, they may wish to share the same set of assumptions about the plans of other aircraft. Such coordination may require time synchronization of planning activities. Each processor then generates plans only for aircraft under its control and communicates its intentions to the other processors. The other processors then replan, using the new information. Deadlocks (inability to find a set of compatible actions) and looping (repeated communications) may occur because of the individual and sequential nature of this planning technique.

In asynchronous cooperative planning, as in iterative local planning, nearby processors exchange goals and plans. However, they also share partial plans, so that discrepancies among intentions become known and can be acted upon quickly. Rather than iterate individually toward a systemwide solution, the planning processors act as a committee, sending, receiving, and revising intentions until an overall solution is found. This technique characterizes many centralized problem-solving architectures (Opplan, Hearsay-II, HASP) and has been successful in distributed AI applications (Lesser and Erman, 1980).

Finally, in simultaneous global planning, processors initially exchange their status and goals, then each autonomously works toward a single overall plan. Each processor attempts to plan for all aircraft, rather than just those under its immediate control. Different plans are generated from the different perspectives of each processor in the group, and bargaining or authority protocols guide selection among alternative solutions.

*Plan evaluation* is closely coupled with conflict recognition. The evaluator tests plans by simulating plan execution, noting the expected conflicts, fuel used, number of commands issued, and schedule delays. The evaluator then aggregates these attributes to determine an overall utility for the candidate plan, the degree of certainty in the evaluation, and the assumptions about other aircraft used in the evaluation. It determines whether the plans meet certain minimum standards, and it selects the best of those that qualify. If none of the plans is satisfactory, the evaluator posts the reasons for their failure.

*Message passing* then is initiated by the communicator, which may request other processors to amend their plans, announce changes in its own plans, or transmit sensed changes in the environment deemed important to other processors. To accomplish these tasks, the communicator must have procedures for checking the status of the communication channels, formatting and transmitting messages, monitoring transmissions, receiving messages, and decoding them. Decisions about whether to send a particular message require the use of an information value model that considers the needs of the sender and recipient and the cost of communication.

*Bargaining* resolves disagreements concerning goals and consequences. The processors involved in a conflict may have different goals, since they compete for the use of limited resources (e.g., space, fuel) and desire efficient routings. When conflicts over the use of these resources cannot be easily resolved, the processors

must bargain or designate an arbitrator. In addition, the processors involved in a conflict may agree on goals but disagree in their estimates of the consequences of actions. For example, one processor might estimate that a given set of plans will result in 40 miles of vectoring for the group, while another processor might estimate 30 miles in detours. Again, some form of bargaining or designation of an arbitrator is necessary to arrive at a consensus.

This bargaining can impose heavy communication requirements on a system of distributed processors. With N processors, as many as $N \cdot (N - 1)/2$ communication links are necessary. If extensive bargaining is required, the process may involve many rounds of iteration and message exchange. Some savings in communication can be achieved by designating an arbitrator. For example, the various processors could send their candidate plans to the arbitrator, who would compare plans, using a "social welfare function" (Farris and Sage, 1975), and select the "best" plan. The arbitrator would then send the selected plan and instructions for it back to the various processors. This structure requires at most $N - 1$ communications links.

Finally, *plan execution* involves the execution of plan commands and the monitoring of plan success by the controller. During plan execution, if any assumptions on which the plan was based are violated, the controller posts a request for replanning in the world model.

These activities constitute an initial set of building blocks for defining distributed problem-solving organizations in ATC. Several forms of coordination among processors are possible. Processors may plan independently, without communication, or they may communicate to exchange data, plans, and constraints. They may plan only for their own aircraft's actions or they may attempt to find a solution for all interacting aircraft. They may bargain over their differences or they may defer to the decisions of an arbitrator or decisionmaker. In the next section, we examine a set of specific problem-solving organizations that incorporate each of these forms of cooperation.

# III. ARCHITECTURES FOR DISTRIBUTED AIR TRAFFIC CONTROL

In this section, we propose six distinct organizations, or architectures, of multiple processors for cooperative planning. Each provides a complete structure for sensing situational conditions, sharing information, recognizing problems, planning cooperatively, and coordinating execution. After describing the structure and characteristics of the six architectures, we illustrate their application in several typical ATC environments.

The architectures, summarized in Fig. 2, use three distinct distribution methods. The first exploits a natural association of processors to the objects they control or the data they gather. These individual-planning methods are based on objects that develop plans for themselves and coordinate with the plans of others. The second type of distribution takes advantage of clusters suggested by the planning environment, such as groups of aircraft in high-density regions or aircraft clustered during approach to an airport. The third type of distribution focuses on decomposition of the solution process, rather than the environment, to distribute planning effort. Here a complete solution must be found by one of the processors. The processors search different options by starting at different points or by using different information.

The object-centered autonomous and object-centered cooperative architectures are variants of a one-processor-per-vehicle design. This one-to-one design is important because it has an increasing number of applications to onboard processing. Collision avoidance systems, cruise missile guidance systems, and automated landing systems all demonstrate the advantages of providing sensing, processing, and planning capabilities in the controlled vehicle. The autonomous and cooperative variants of the object-centered structure reflect different styles of distributing planning functions among the controlled vehicles. The autonomous structure relies on each processor's own sensing and inference for information-gathering—planning and plan execution occur without communication among processors. In the cooperative structure, aircraft communicate to exchange data and collaborate during planning.

The space-centered and function-centered architectures derive from situations with natural, stable groupings of objects, most of which occur along either regional or functional lines. In a regional grouping, a processor controls all objects within a specified spatial region. Current enroute control of civilian air traffic utilizes a space-centered architecture. In a functional grouping, each processor controls all objects engaged in a particular type of activity. Distribution by function frequently occurs in terminal control areas, where different controllers handle approaches, landings, and overflights.

The final two architectures, plan-centered and hierarchical, address problems requiring a global solution—that is, one that considers all aircraft. A problem requires a global solution (rather than a combination of local partial solutions) if interactions affect all objects in the airspace, or if environmental dynamics preclude stable groupings. The plan-centered structure assigns to each processor a different

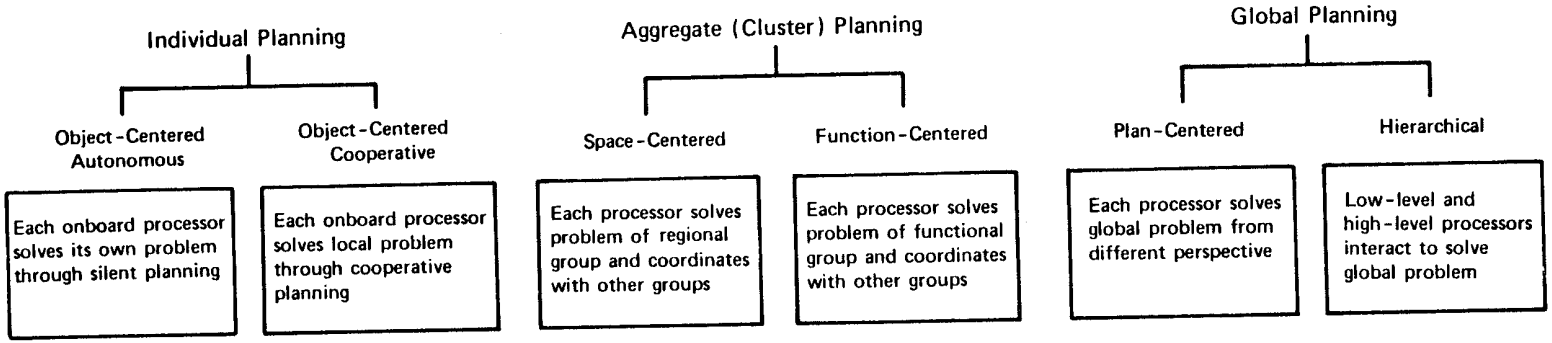


Fig. 2 — Architectures for distributed planning

portion of the search space or a different approach to the problem. The processors work silently until one arrives at a solution. This type of organization is exemplified by oil exploration and drilling. The hierarchical approach attempts to achieve a global solution by decreasing the problem into high-level decisionmaking and low-level sensing and problem-solving. The high-level processors rely on global data aggregation and extensive communication. Low-level processors execute plans, monitor plan execution, and perform local replanning. We describe each of these architectures and distribution methods in more detail below.

## OBJECT-CENTERED AUTONOMOUS ARCHITECTURE

The object-centered autonomous architecture is a communication-free organization in which each aircraft performs all situation assessment, conflict recognition, planning, and control functions autonomously. Figure 3 illustrates a possible event sequencing among three aircraft using the object-centered autonomous architecture. In this case, planning is reactive and incremental, and preplanned protocols are used for resolving conflicts. Natural organizations that exemplify this type of autonomous planning and control include joggers, who avoid collisions by independent prediction and response, and automobile traffic at unregulated intersections.

Figure 4 shows the organization of activities for each aircraft in this architecture. This diagram is a variant of the problem-solving kernel shown in Fig. 1. The major difference is the absence of the communicator module. The inability to communicate places a greater burden on the sensor to infer the plans of others from sensed locations, altitudes, and headings. Since these estimates can be incorrect and uncertain, plan evaluations can only be tentative and must be frequent. Replanning may be required.

For example, suppose that a slow propeller craft in level flight is potentially in conflict with an ascending jet, as shown in Fig. 3. Each aircraft may sense the other and use trajectory projection to predict the time and location of the conflict. Each aircraft may use its plan generator to synthesize possible maneuvers for avoiding the other. The plan generator posts the selected action in the world model for the evaluator to test using a fast-time simulation. The simulation returns the predicted conflicts and expected effects on fuel and schedule. If the expected performance is acceptable, the controller executes the plan. If the expected performance is unacceptable, the plan generator produces a different action sequence, using information about why the previous actions failed.

Since this architecture allows consideration of only single-aircraft actions, problems may arise when actions selected independently by the aircraft are incompatible or inefficient. Incompatibility results, for example, if both aircraft decide to detour in the same direction. The ability of the system to iterate safely to a conclusion depends to a large extent on tracking delays and sensor accuracy (Andrews and Hollister, 1980) or on a complete set of "rules of the road." Efficiency problems result if both aircraft maneuver when only one needs to, or if the aircraft use overly conservative separation requirements and resolution lead times. The aircraft need such "cushions" only when they have incomplete data regarding the intentions of other aircraft.

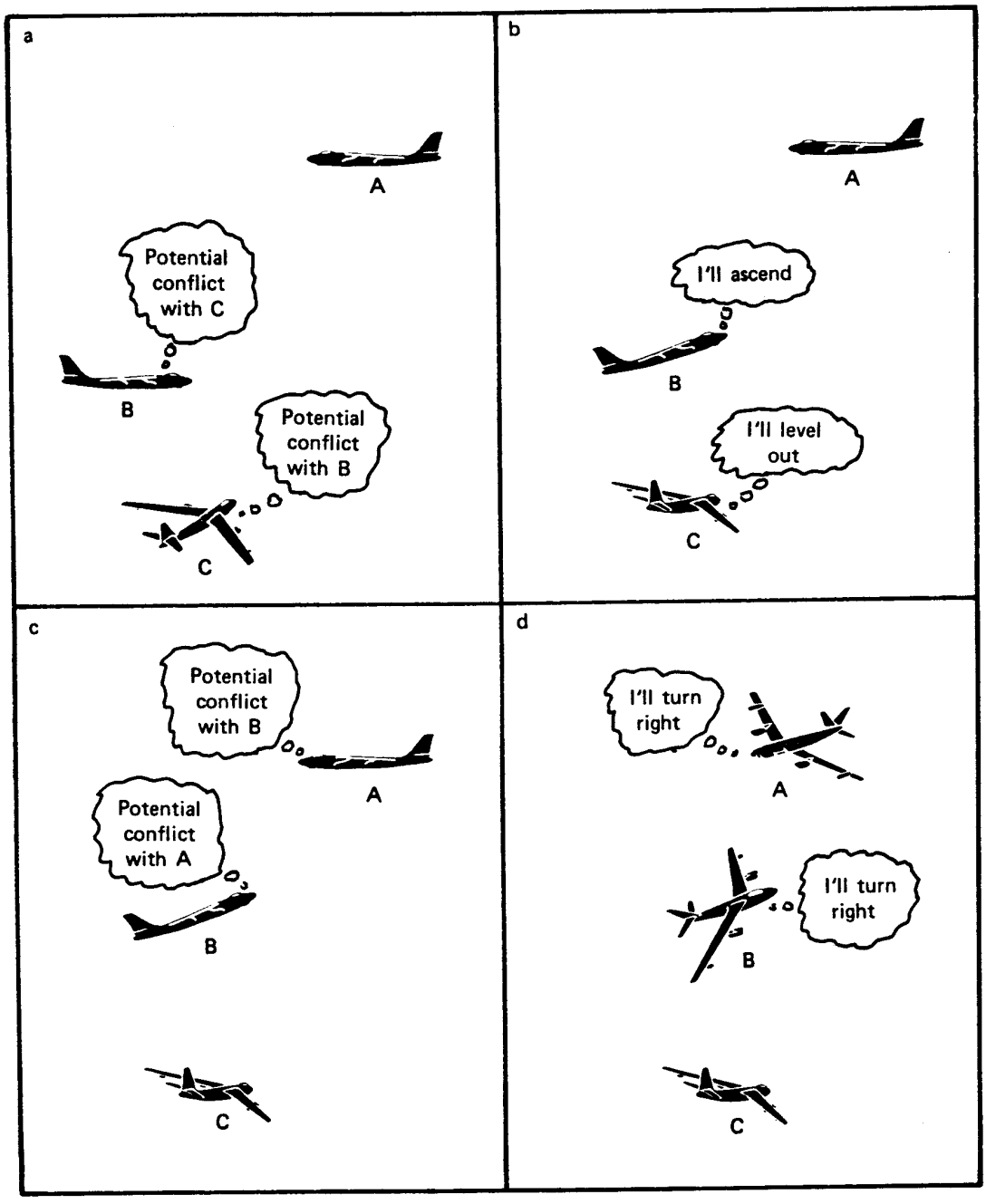


Fig. 3 — Illustrative sequence of interactions in the object-centered autonomous mode

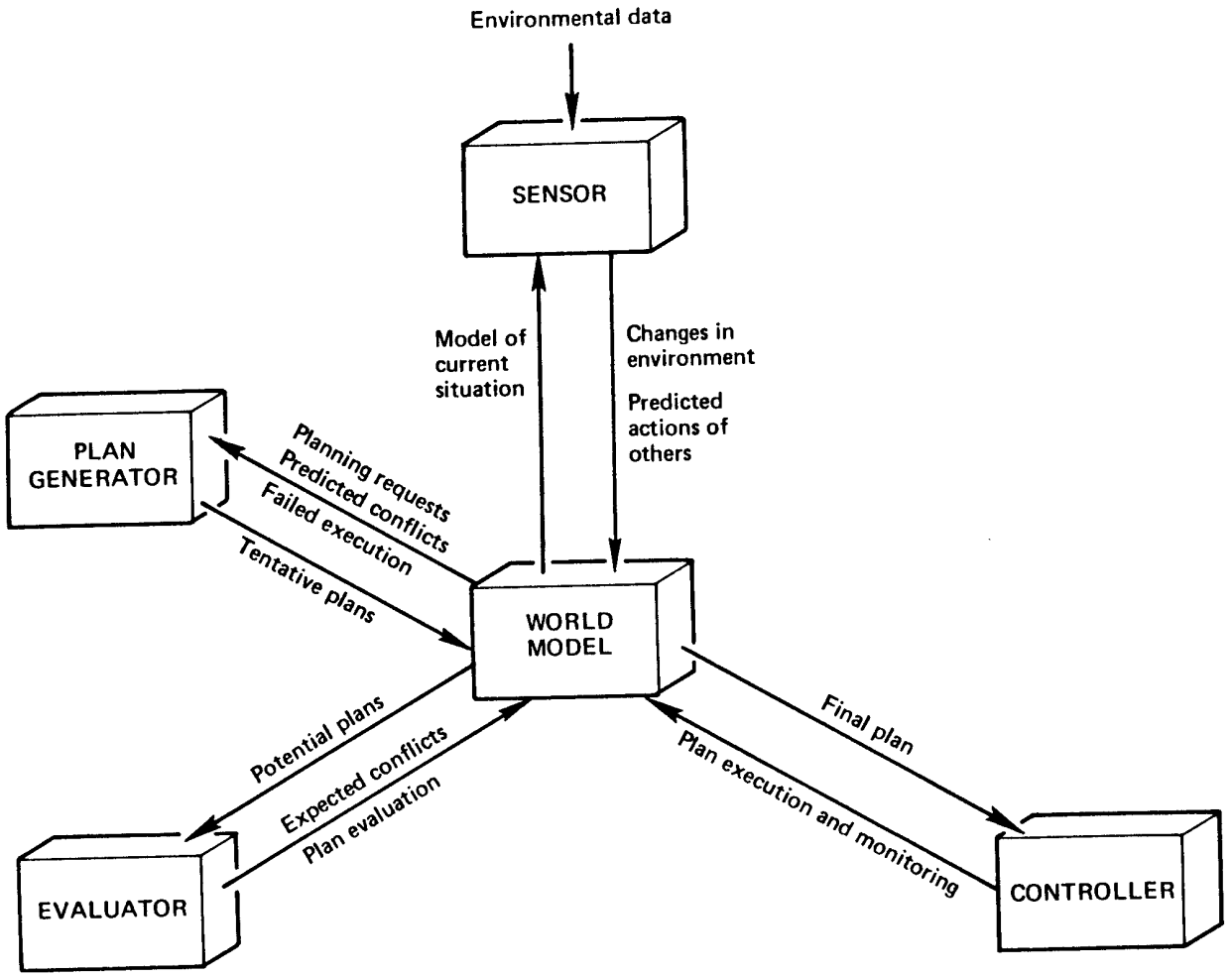


Fig. 4—The structure of a processor in the object-centered autonomous architecture

## OBJECT-CENTERED COOPERATIVE ARCHITECTURE

The object-centered cooperative architecture also has a processor associated with each aircraft, but the aircraft communicate to plan collaboratively. Figure 5 illustrates how neighboring aircraft might plan cooperatively to resolve conflicts. In this case, new plans are developed and agreed upon prior to execution.

The processing capabilities associated with each aircraft processor now expand to include communication procedures, as shown in Fig. 6. If communication costs are low, knowledge that might have been derived through inference can instead be obtained though communication with the aircraft in question, as shown by the activities of the communicator.

Decisions about whether to communicate or to perform computations locally depend on the expected value of the information and the relative cost of obtaining it by either method. Unfortunately, accurate computation of communication cost requires that the sender have accurate knowledge of the recipient's goals, world model, and pending activities. In general, communication is more desirable than local inferencing only if (1) the information changes the current situation assessment of the recipient, (2) the new situation assessment leads to a different action than was originally planned, and (3) the expected value of the new action is greater than the expected value of the previously pending action plus the cost of transmission (Marschak, 1973).

The planning process changes dramatically with the addition of communications. In the example of the slow propeller craft conflicting with an ascending jet (illustrated in Fig. 5), a new set of options emerges. The propeller plane may generate and evaluate maneuvers for itself, for the jet, or both. If the fast-time simulation in the evaluator indicates that a slight change in the jet's trajectory results in the least overall fuel usage and the greatest safety, the propeller aircraft can request such an action via message-passing.

This architecture can support either iterative local planning or asynchronous cooperative planning. Potentially conflicting aircraft exchange goals and constraints and then formulate, send, revise, and receive partial plans until a global solution is achieved. In this way, aircraft plan in parallel, occasionally sharing information to focus their efforts and prune unpromising alternatives. Since plans are shared, processors may use accurate plans of others in their fast-time look-ahead. The frequent short-distance communications used in the object-centered cooperative architecture favor the use of a flexible local-area network in which the aircraft use broadcast transmissions to locate other aircraft and point-to-point transmissions to make the data transfers (Clark, Pogram, and Reed, 1978).

The inherent redundancy of the object-centered cooperative architecture should make it relatively immune to losses of individual processors or breaks in communication links. Loss of a processor or communication link for some interval can be detected by the other aircraft through monitoring of communication exchanges or from inference following unexpected behavior. The other aircraft should be able to make plans that accommodate the degraded performance of that processor. Needed data may be requested from other aircraft or obtained from the degraded aircraft by rerouting transmissions through links that are still secure. Simple data inaccuracies may be recognized and treated through comparison of redundant data structures onboard each aircraft.

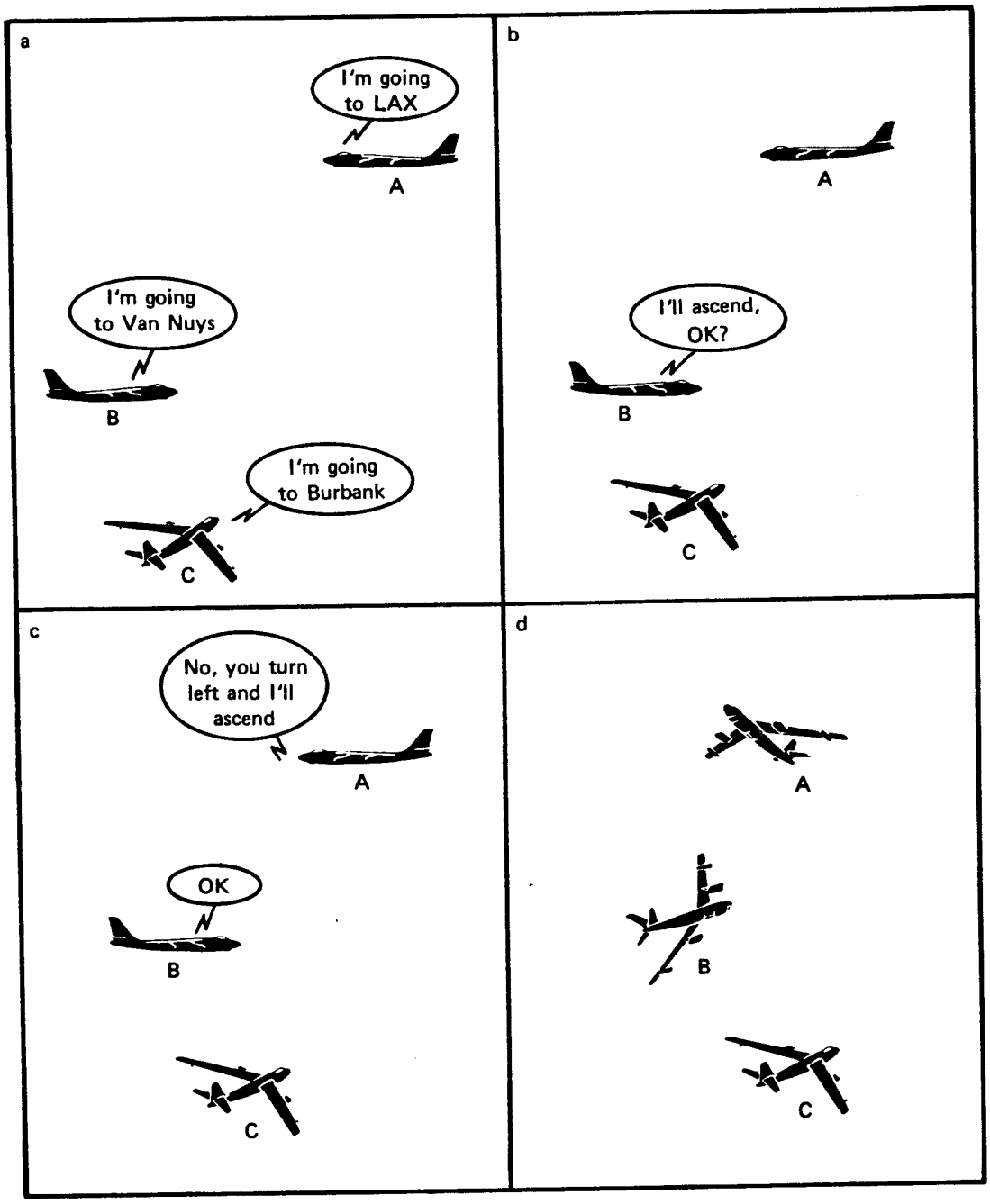


Fig. 5—Illustrative sequence of interactions in the object-centered cooperative mode

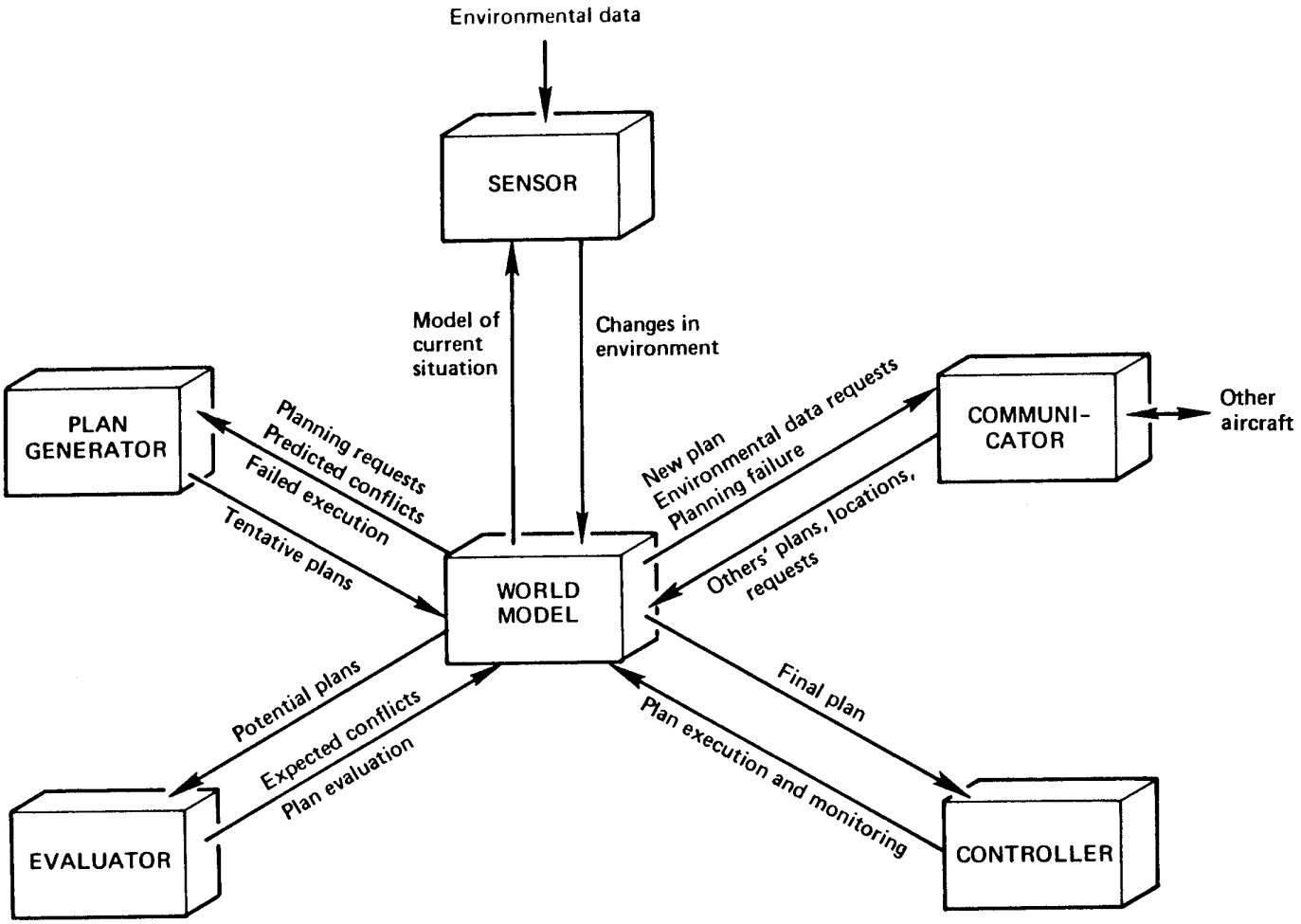


Fig. 6 —The structure of a processor in the object-centered cooperative architecture

On the other hand, the object-centered cooperative architecture is not suited to long-range strategic planning or complex multi-aircraft interactions because of its small scope of view. This architecture appears most effective in situations having range-limited communications and time-stressed decisionmaking. It minimizes long-distance communications, each aircraft has maximum autonomy to respond to unforeseen conditions, and planning loads are evenly distributed throughout the group.

## SPACE-CENTERED ARCHITECTURE

Current ATC facilities exemplify the space-centered architecture, in which each processor controls a region of space rather than a particular aircraft. Figure 7 illustrates a typical interaction within this structure. A ground-based or airborne processor monitors and controls all aircraft within an assigned region. Sensor data collected by individual aircraft or by special equipment (e.g., ground-based radar) are aggregated at the control site for use in planning. Since the sector center has plans for all aircraft, this architecture (like the object-centered cooperative one) can use the plans in its fast-time look-ahead. The planning program can plan globally for the entire sector, or it can define independent conflict clusters and plan within these separately.

The communications processes, shown in Fig. 8, are quite different from those of the object-centered architecture. Control centers communicate with both individual aircraft and other control centers. Control centers receive sensor data from and transmit commands to aircraft in their sector. Communication between centers consists of data, plans, and action requests about activity at the sector boundaries. Accordingly, the communication system must contain rules for prediction of the boundary crossing point, for bargaining about the point of transfer of control, for transfer of data and plans concerning the aircraft, and for acknowledgment of control transfer. Occasionally, adjacent sectors will also coordinate to achieve flow management (i.e., delay takeoffs or slow outbound flights to smooth out load peaks).

This hand-off process can be extremely complex. Typically, handoffs are governed by preplanned protocols that prescribe locations, altitudes, and bearings for transfer of control. Preferably, however, a transfer of control should be considered only if the receiving sector is not overloaded, if the communication channels are reliable, and if the information-transfer time interval is shorter than the allowable decision time. Hand-off rates and locations can be varied, using iterative local planning to achieve load balancing between sectors. For example, where there are large load variations between sectors, the sector boundaries could move dynamically according to the load level. Such boundary movements could be cumbersome and could be potential sources of error, however, so such allocation of responsibility might be best suited to long-term load shifts rather than to environments with rapid load fluctuations.

We expect the space-centered architecture to be most effective in terminal and enroute areas with reliable, high-bandwidth communications, complete radar coverage, and relatively constant and predictable load levels. The space-centered architecture uses centralization of data fusion and planning to facilitate problem-solving on a larger scale than is performed in the object-centered architectures. Unfortu-

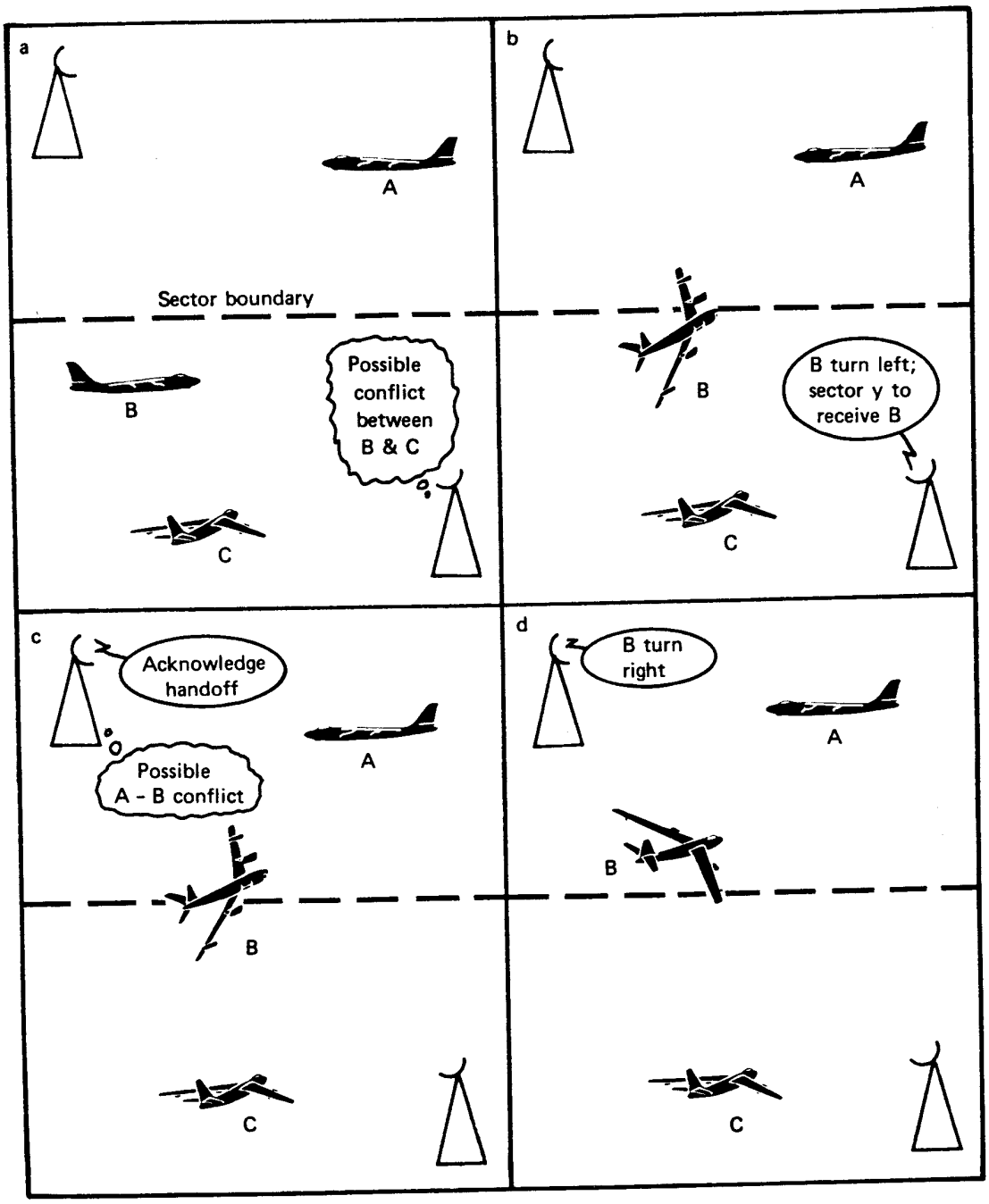


Fig. 7—Illustrative sequence of interactions in the space-centered mode

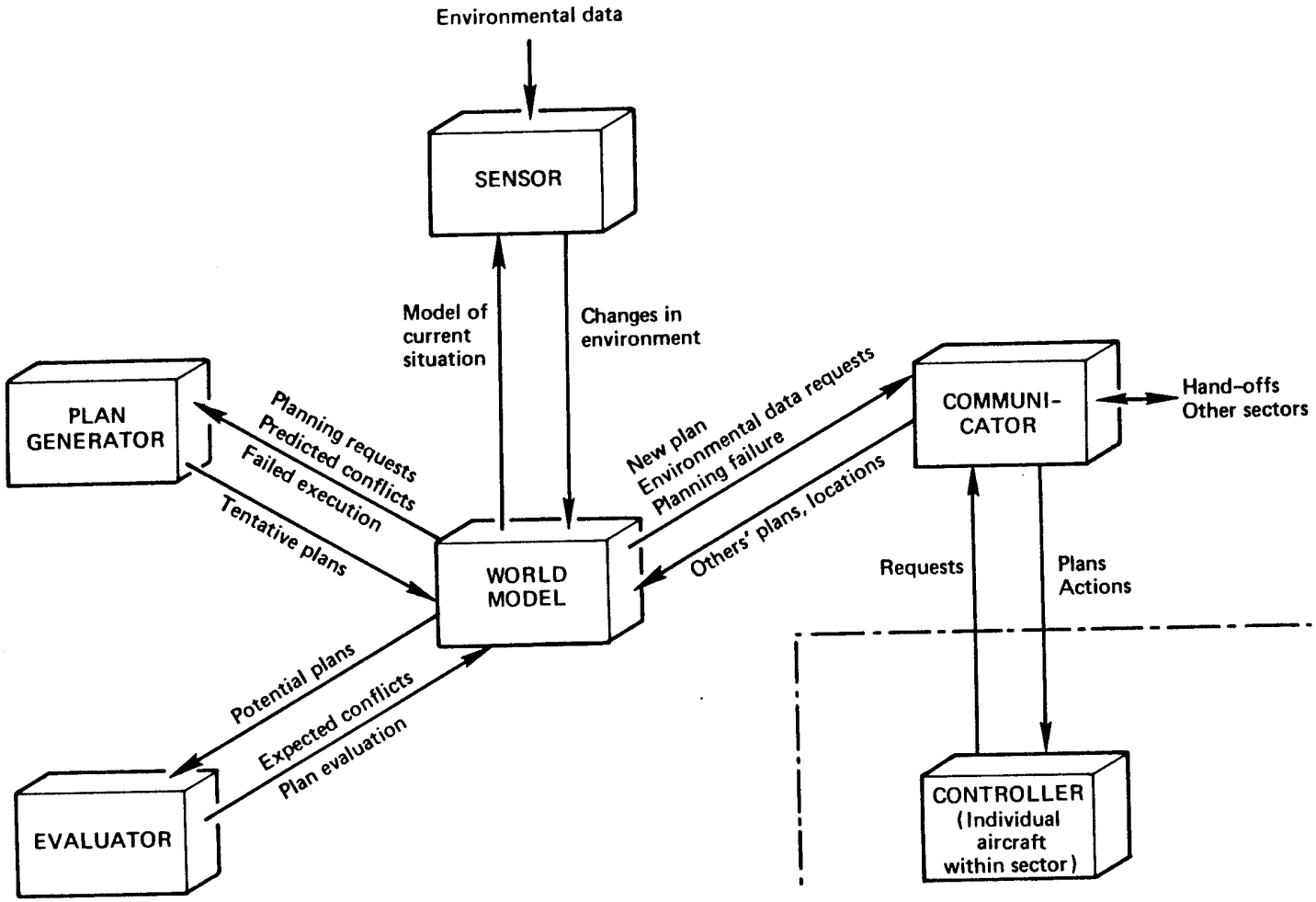


Fig. 8—The structure of a processor in the space-centered architecture

nately, the architecture must be supported by extensive processing and communications networking capabilities, and it is vulnerable to communication system or processor losses.

## FUNCTION-CENTERED ARCHITECTURE

The function-centered architecture, like the space-centered approach, assigns multiple objects to each processor. A processor may control all aircraft within a single flight phase (e.g., takeoff, transition, enroute, or approach) or may be responsible for a specific type of aircraft (e.g., private, commercial, or military). Each processor controls several aircraft in a manner similar to the space-centered architecture illustrated in Fig. 8, with one exception: Hand-offs occur between processors that control different functions within a geographic region rather than different regions. As with the space-centered architecture, cooperation is achieved either through the use of preplanned protocols or iterative local planning.

Assignment of aircraft to processors in the function-centered architecture can occur on the basis of time, location, or load. Thus, allocation of responsibility and coordination of effort in the function-centered architecture may be difficult to achieve and maintain. Also, the geographic overlap of groups controlled by different processors means that some inter-processor interactions will involve aircraft conflict resolution, and some will involve control hand-off. As a result, the function-centered architecture shares many of the problems of both the object-centered cooperative and the space-centered structures.

Nevertheless, the function-centered architecture seems well suited to the coordination of traffic near terminals, since current ATC operations in the terminal areas tend to cluster aircraft into natural functional groups. This architecture concentrates data fusion and planning responsibility at a small number of sites, thus facilitating long-range multi-aircraft planning. However, it is not as applicable to communications-limited situations, since it requires high-bandwidth, long-distance communications. It is also vulnerable to communication and processor losses because of its concentration of data and knowledge at a few sites.

## PLAN-CENTERED ARCHITECTURE

The plan-centered architecture, portrayed in a terminal-area ATC situation in Fig. 9, distributes the planning process by assigning different approaches or portions of the search space to each processor. This architecture is suited to domains characterized by relatively simple problem-solving requirements and a problem space in which solutions are very sparse. In such cases, each processor can attack the entire problem but can explore only a portion of the entire search space. All processors attempt to find a solution to the overall problem using simultaneous global planning, but they do so independently. An illustrative ATC situation, shown in Fig. 9, would be the impending convergence of several aircraft in a highly congested airspace. Each processor, acting independently, would seek a unique resolution to the conflict by electing, for example, to alter the routes of a different subset of conflicting aircraft.

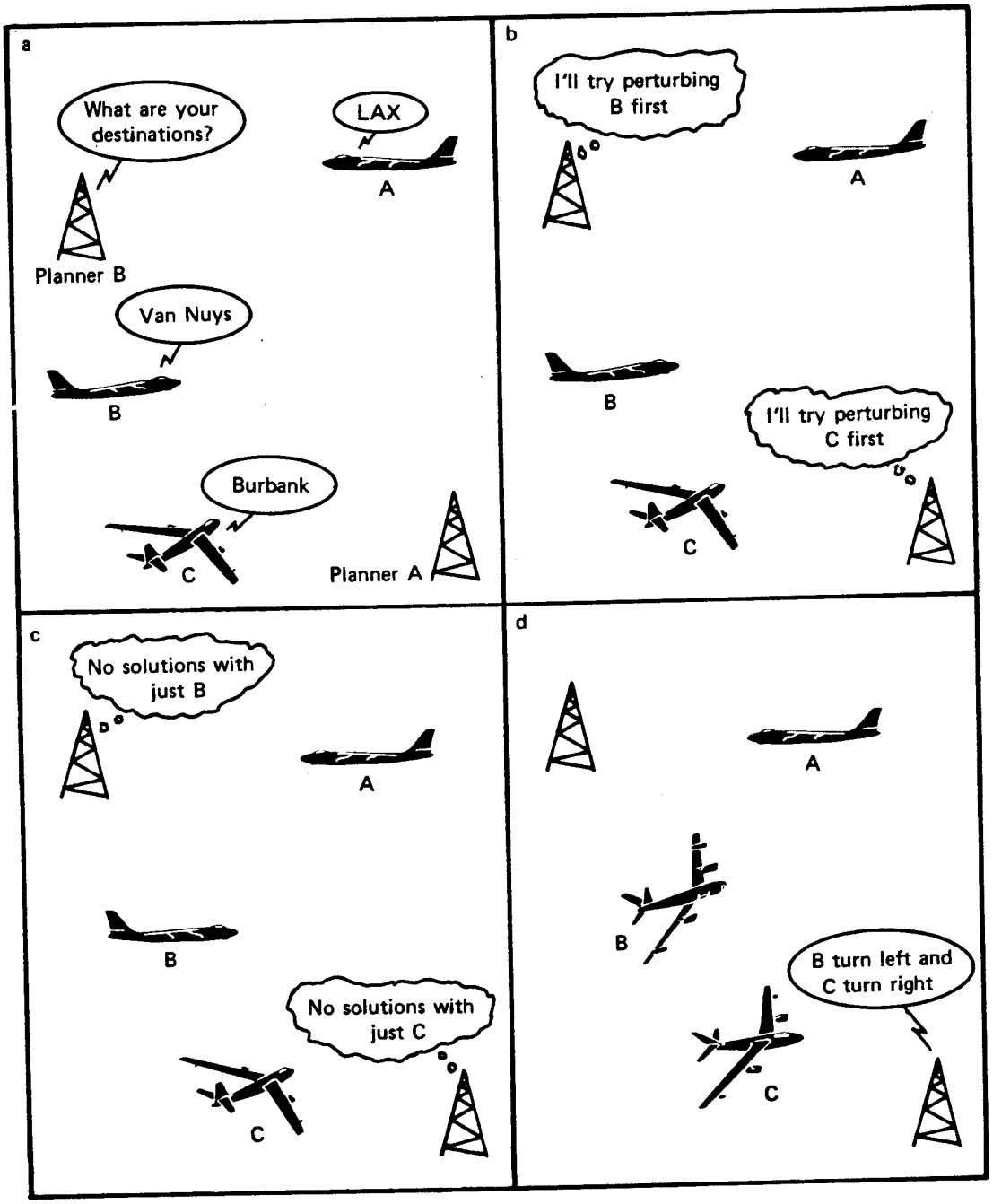


Fig. 9—Illustrative sequence of interactions in the plan-centered mode

Figure 10 presents the structure of a plan-centered node. In performing simultaneous global planning, each processor requires accurate knowledge of the locations and intentions of all aircraft. Thus, processors exchange data via communication to perform situation assessment. No cooperative planning takes place, however, and the amended plans are transmitted only if a processor discovers a suitable global solution.

The relatively high overhead incurred by extensive knowledge distribution makes this architecture suitable only for complex, high-density ATC situations. The plan-centered architecture also appears appropriate in situations with frequent processor losses, since each processor essentially acts as a redundant element.

## HIERARCHICAL ARCHITECTURE

Some problems are inherently hierarchical in structure. In a hierarchical architecture, lower-level nodes gather information and/or control objects directly. They pass abstracted and aggregated information up the hierarchy to supervisory nodes, which both direct their subordinates' behavior and report to their own supervisors. A high-level decisionmaker postulates and evaluates strategic plans. A hierarchical architecture most closely resembles the structure of a centralized problem-solver, in that the upper-level nodes have approximately global perspectives on the problem.

Cockpit display of traffic information (CDTI) scenarios postulated by Lincoln Laboratory (Andrews and Hollister, 1980) and by NASA-Ames Research Center (Kreifeldt et al., 1976) exemplify hierarchical ATC operations. In these proposed systems, the ground control center acts as the high-level supervisory controller, while the CDTI-equipped aircraft perform self-separation and short-time-horizon route planning.

The local and supervisory nodes in the hierarchical architecture are shown in Figs. 11 and 12. The local processor functions similarly to a processor in the object-centered cooperative structure. Instead of communicating only with other equivalent processors, however, the local processor communicates with the supervisor. The local processor senses and aggregates data locally, performs local planning, sends abstracted data to the supervisor, and executes commanded actions. The supervisor node, much like a space- or function-centered node, collects and processes the data sent from the local processors, performs long-range planning, and sends commands back to the local nodes.

This hierarchical architecture entails high overhead in communications. To solve a problem, information and requests must always be passed up the hierarchy to centralized decisionmakers (Parnas, 1974). Furthermore, information exchanged among low-level nodes must often be routed through intermediate-level "managers," thus increasing communication time and communication loads (Wesson et al., 1980). The hierarchical architecture appears most appropriate for applications with natural levels of abstractions (such as flow control and separation assurance in terminal-area ATC), high-bandwidth and relatively error-free communications, and a reliable supervisory processor.

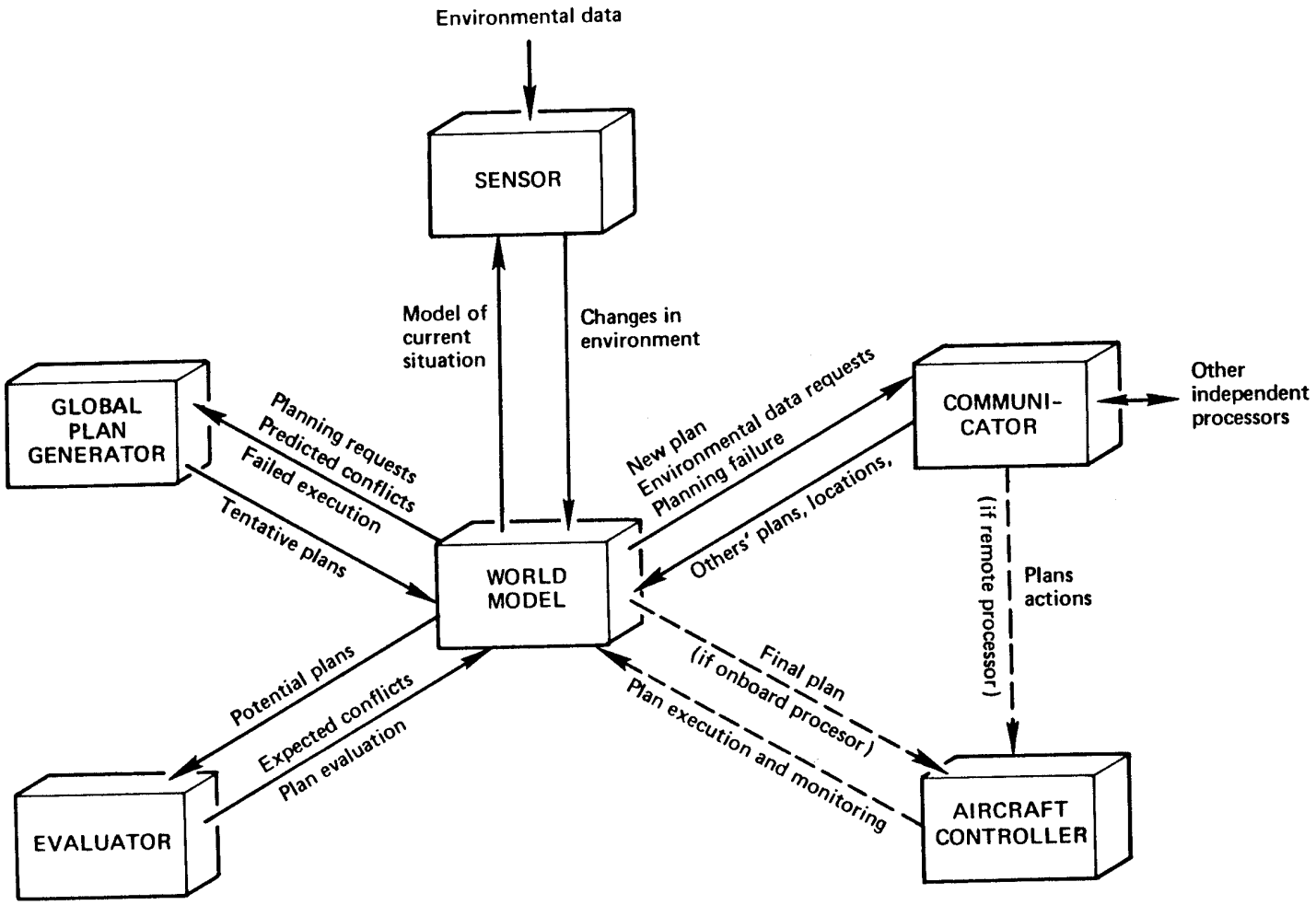


Fig. 10 —The structure of a processor in the plan-centered architecture

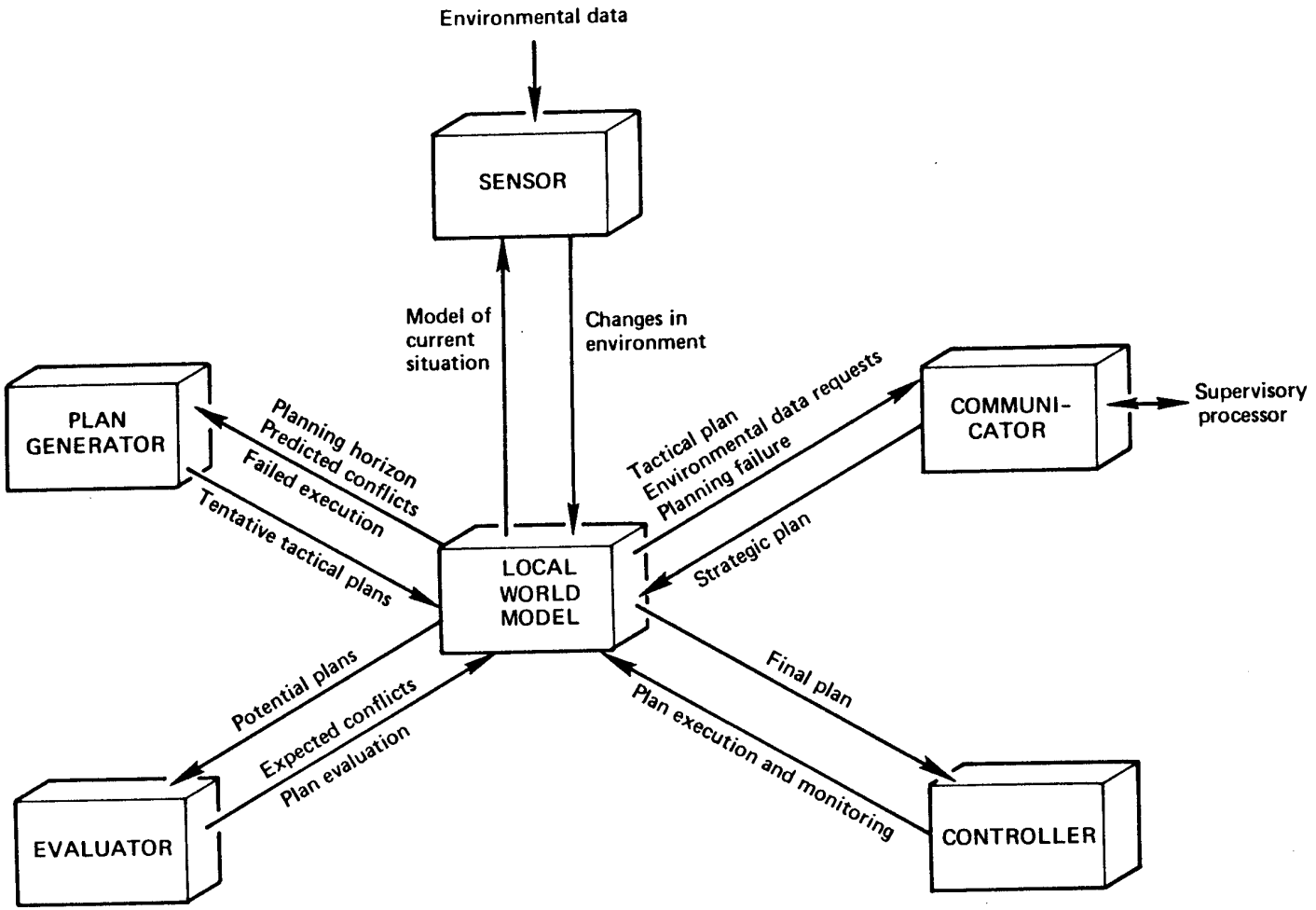


Fig. 11 — The structure of a local or low-level processor in the hierarchical architecture

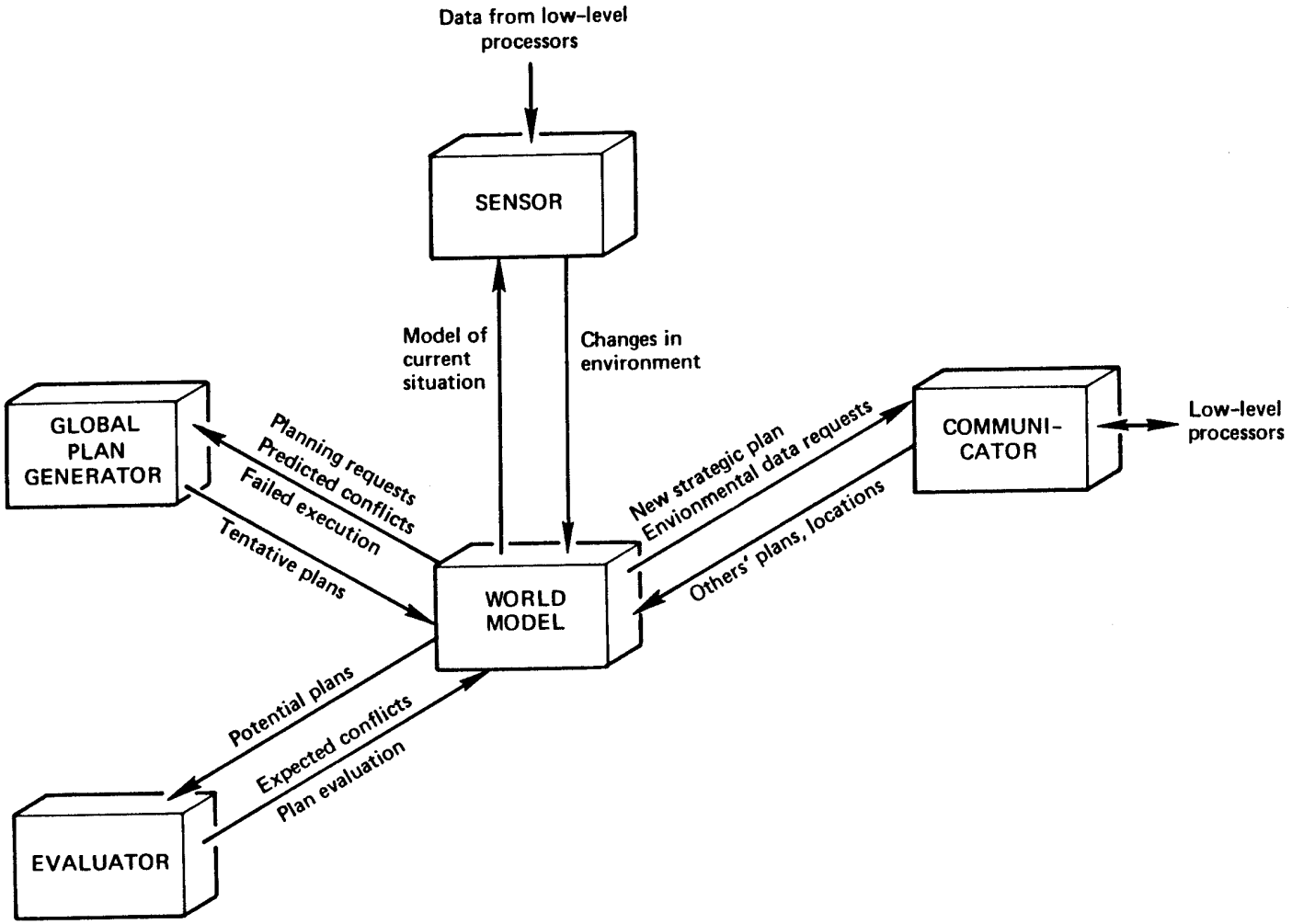


Fig. 12—The structure of a supervisory or high-level processor in the hierarchical architecture

# IV. ENVIRONMENTAL INFLUENCES ON COOPERATIVE PROBLEM-SOLVING

In Section III we examined general relationships between distributed problem-solving architectures and the task environment. In this section, we consider potential task environments in more detail. In developing each of the architectures, we made certain assumptions about the characteristics of the task and the processors cooperating to perform the task. In reality, a variety of environmental constraints may confront designers of distributed systems. Therefore, we must evaluate the utility of the different architectures and problem-solving methods under a variety of environmental conditions.

Our investigations of air fleet control revealed eight important environmental attributes:

1. *Environmental uncertainty*—the prior uncertainty of environmental conditions, aircraft locations, and aircraft intentions.
2. *Environmental dynamics*—the variability of environmental conditions over time.
3. *Communication constraints*—the range, noise level, and bandwidth limitations of available communications.
4. *Degree of clustering*—the extent of functional grouping of the processors.
5. *Time stress*—the time available for decisionmaking.
6. *Option multiplicity*—the number of planning options available to each processor.
7. *Density of the solution space*—the ratio of acceptable, conflict-free plans to the number of potential plans.
8. *Situational complexity*—the number of elements necessary for problem representation.

We shall describe each of these environmental conditions and how they influence the choice of cooperative problem-solving architectures and methods.

## ENVIRONMENTAL UNCERTAINTY

The world model may not contain accurate or certain knowledge of crucial environmental states—traffic, weather, terrain, etc.—that the plan generator and evaluator use in formulating plans. Such situations occur, for example, in general aviation in mountainous areas. Methods to deal with this problem include the following:

1. *Data pooling.* Processors may integrate sensor data at designated nodes to reduce data uncertainty. If situation assessment requires aggregation of data from most or all possible sources, a hierarchical structure may be optimal. Unfortunately, a hierarchical structure may result in complex and time-consuming calculations, since complete situation assessment en-

tails continuous updating of airspace conditions and aircraft locations in a large region.

2. *Data quality evaluation.* The representation of world knowledge should incorporate certainty levels that reflect sensor accuracy, channel quality, recency of sensor reports, and reliability of the knowledge aggregation method. These values should be revised in successive situation updates and used to guide the search for either robust plans or alternative plans to implement in the event of possible environmental changes.
3. *Expansion of the planning function to include data-gathering options.* When possible, the planning choices should include the option of delaying actions and gathering further data. This requires the use of an information value model that weighs the usefulness of the information against the time delays and costs involved in obtaining it.
4. *Reduction of look-ahead horizons.* The conflict-detection and plan-evaluation routines of the evaluator must make shorter projections into the future. Because of the cascading of uncertainties with time, problem recognition, planning, and evaluation based on short projections may be more effective than those based on long projections.

## ENVIRONMENTAL DYNAMICS

Task conditions (e.g., geographic region, data links, traffic loads) may vary with time. This problem is closely related to that of environmental uncertainty, particularly if the dynamics of the environment are unpredictable. Induced architectural refinements include:

1. *Transitions among architectures.* Distribution schemes relying on stationary communication links, standard protocols, or predictable sector loadings may become handicapped in dynamic environments. Such schemes include the space-centered, function-centered, and hierarchical architectures. The more flexible architectures should be invoked to distribute planning and control dynamically as demanded by the changing situation.
2. *Maintenance of belief or confidence estimates.* Situation assessments should have associated confidence and perishability estimates. To monitor the accuracy of these beliefs, the system should maintain data dependencies between linked beliefs, as is done in Hearsay-II (Erman et al., 1980).
3. *Dynamic communications addressing.* The architecture must support continuous communication between the nodes. Communication demands between nodes may be unpredictable, as well as non-uniformly distributed in time. This may require use of broadcasts to locate nodes and point-to-point communications to pass messages.
4. *Increased inference capabilities.* The situation assessment and planning mechanisms may have to rely on inexact pattern-matches to recognize conditions in the unpredictable environment. This may require inference systems that can partially match situation descriptions to rule antecedents (Hayes-Roth, 1978).

## COMMUNICATION CONSTRAINTS

The communication links between aircraft may be degraded as a result of bandwidth limitations, environmental interference (e.g., mountains, weather), or range limitations. These problems could be alleviated by the adoption of additional types of sophistication:

1. *Increase in local area networks.* Each separable cluster of aircraft may have to rely more on specialized local area networks than on long-distance communications to remote processing centers. Clark, Pogram, and Reed (1978) discuss the advantages of such local-area networks in geographically limited communication tasks.
2. *Network flexibility.* Because of the possibility of blockages or losses, the networks must have alternative pathways for transmission. ARPANET-like protocols may be used to check link conditions and specify back-up routings (Kahn et al., 1978). Network flexibility may also be increased by multi-hop routing, i.e., routing the data in several hops through the fleet members. This may reduce the individual link distances and link data rates, as compared to direct broadcast communications, but at the same time it may increase the time delays. Variable network geometries based on packet radio techniques, for example, timeshare digital radio frequencies and are organized so that messages reach their destinations through varying, irregular pathways. Unfortunately, systems with many store-and-forward relay nodes can experience high levels of contention and queuing during peak loading periods (Martin, 1977).
3. *Shift to autonomous problem-solving.* Communication range constraints may force a change from global updating of situation estimates and exhaustive search by each node to a less data-intensive strategy. Local data-sensing, planning, and execution should decrease the amount of long-range high-bandwidth data exchange. This type of organization is found in the object-centered cooperative architecture. In extreme situations, requirements for radio silence may dictate the use of the object-centered autonomous architecture.

## DEGREE OF CLUSTERING

The form of distribution of planning and control depends strongly on the dispersion of aircraft and the location of objects in the environment. When aircraft group naturally into clusters, with virtually all interactions within the clusters, each group of aircraft may be handled separately. The organization of aircraft into clusters significantly reduces the computation required to detect conflicts. (Calculations illustrating this reduction from an exponential problem to a nearly linear problem are presented in the Appendix.) Other changes that may be induced by clustering are:

1. *Knowledge distribution by specialization.* Communication, data fusion, planning, and control may be replicated only within each specific group. This minimizes unnecessary knowledge replication across all processors.

2. *Local area networks.* If the environmental situation is stationary over time, each cluster may require its own local communication network. This circumvents the need for establishing costly, time-consuming procedures for coordinating with other groups.

## TIME STRESS

The time available for decisionmaking depends on aircraft speed, communication constraints, and sensor ranges. In both enroute and terminal-area control situations, problems arise that require resolution within a few seconds. Time stress may require that the architecture have the following characteristics:

1. *High communication channel throughput.* Transmission time in the architectures relying on communications may be reduced through use of very high channel frequencies (Kahn et al., 1978), and response time may be improved by the use of frequency-division multiple-access (FDMA) techniques, in which each link uses a different channel. This avoids the data queues present in single-channel time-division multiple-access (TDMA) systems (Martin, 1977).
2. *Extensive knowledge distribution.* The inclusion of comprehensive knowledge bases in each node processor can reduce the need for interrogating other nodes. Of course, the most critical knowledge may be temporally variable and local in influence. Such information cannot be preloaded into the knowledge bases.
3. *Emphasis on heuristic planning techniques over exhaustive search.* Heuristic planning techniques such as condition-action rules (Newell and Simon, 1972), opportunistic planning (Hayes-Roth and Hayes-Roth, 1978), and simulation-based look-ahead planning (Wesson, 1977) may provide rapid-response capabilities by relaxing the requirement for optimality.

## OPTION MULTIPLICITY

The number of possible actions open to each aircraft increases with the number of maneuver options, the sensing range, and the physical extent of the available airspace. The complexity of the planning problem suggests the following problem-solving approaches:

1. *Heuristic planning techniques.* Complete and exhaustive search for optimal routes may be too time-consuming. Heuristic search methods can prune from consideration many planning options, thereby minimizing the number of options to be evaluated in computationally expensive fast-time simulation algorithms.
2. *Planning distribution.* Distributing the planning tasks among a large number of parallel processors can exploit the availability of multiple systems. Each processor might adopt a different solution strategy or take responsibility for a different function or geographic area. The most extreme examples of this approach are the object-centered autonomous and

plan-centered architectures, in which completely disjoint sets of options are considered by each processor.

## DENSITY OF THE SOLUTION SPACE

There may be very few conflict-free solutions in a high-density ATC airspace, because of the large number of routing constraints. The following problem-solving approaches are favored in this situation:

1. *Emphasis on deep search.* Determining one of a very few solutions is similar to finding the optimal solution. The planning program must examine virtually all possible options or use a highly discriminating evaluation function to prune non-optimal paths. Such an evaluation function requires extensive simulation-based look-ahead capabilities.
2. *Global view.* Unless the problem can be decomposed into almost completely independent subproblems, the complete problem must be solved. If the complete problem is confronted, information and knowledge must be concentrated at a single processing center. Alternatively, the overall problem may be distributed by assigning individual branches of the search tree to different processing nodes, as in the plan-centered approach.

## SITUATIONAL COMPLEXITY

Situational complexity is a function of the number of elements that must be represented—aircraft, navigational aids, terrain obstacles, weather conditions, airfields, etc.—and the amount of conflict between the goals of the participants. Both factors increase the demands on data fusion, problem recognition, planning, and bargaining. High situational complexity suggests the following architectural features:

1. *Data distribution.* The communication of sensor data may be limited to those nodes requesting such data. The recipient nodes may be those closest to or most affected by the object under surveillance. Limiting the dissemination of situation data reduces the communication requirements and processing demands.
2. *Designation of an arbitrator.* If the different processing nodes have very different goals, they may have to bargain for scarce resources such as space and fuel. Unstructured bargaining requires communication of candidate plans, pairwise comparisons between all competing plans, and transmission of evaluations between all affected processing nodes. Designation of one processor as an arbitrator for each conflict reduces the maximum number of communication links from $N \cdot T(N - 1)/2$ to $(N - 1)$.

# V. AN ATC SCENARIO: THE OBJECT-CENTERED COOPERATIVE ARCHITECTURE IN THE TERMINAL-AREA CONTROL ENVIRONMENT

This section presents an in-depth analysis of the object-centered cooperative (OCC) architecture, in the context of terminal-area control. This architecture embodies features of many distributed AI situations: multiple processors, conflicting goals, dynamic communication demands, and both autonomous and cooperative planning. We chose the OCC architecture because it exhibits virtually all of the forms of cooperation shown by the other architectures, and it does so with the simplest structure. Terminal-area control was selected because it exhibits difficult problems of communication constraints, time stress, and situational complexity.

The terminal area is an extremely congested traffic hub. As aircraft approach the hub, both the frequency of conflicts and the amount of short-term tactical control increase. Traffic typically is funneled along certain pathways for landings, takeoffs, and overflights. At present, aircraft coordination is accomplished through a combination of ground control, pilot problem-solving, and adherence to established rules-of-the-road.

An object-centered architecture represents a radical departure from this traditional organization. In the OCC architecture, ground control is replaced by a system in which the individual aircraft perform all situation assessment and planning entirely through a combination of silent, autonomous planning and cooperative planning.

Planning a route to maintain separation between aircraft requires a vast amount of knowledge and expertise. Table 2, adapted from Andrews and Hollister (1980), lists some of the required knowledge, portions of which may be obtained from a centralized database; the rest must be acquired from onboard sensing, computation, or communication with other aircraft.

Our system design for the OCC architecture assumes identical processing capabilities in each of the aircraft in the airspace. We presume that each aircraft has a complete view of the the environment and engages in noise-free communications.

Our task environment, an automated ATC simulation, provides a real-time control environment in which one or more air traffic controllers manage an area of airspace containing nine entry/exit points and two airports. The controller(s) must issue commands to 26 aircraft that appear during the course of the simulation, guiding each aircraft to the destination listed on its flight plan. The airspace includes airways that link the entry/exit fixes at the edges of the airspace, airports with designated directions for takeoffs and landings, and navigational aids for use as reference points for vectoring the aircraft. The simulation, implemented on a PDP 11/70 computer, displays a controller's radar terminal on a CRT screen and uses alphanumeric symbols to represent aircraft. The controller(s) must issue commands to the aircraft to route them to their specified destinations while maintaining separation between the aircraft and remaining within fuel limitations.

Table 2

INFORMATION REQUIRED FOR SEPARATION MAINTENANCE

| Information | Application |
|---|---|
| Positions (relative) | Computation of relative motion. |
| Positions (absolute) | Location with respect to terrain, airspace structure, or airfields. |
| Terrain, airspace boundaries, minimum descent altitude | Identifying conflict-resolution options consistent with airspace constraints. |
| Turn rate of aircraft | Flight-path estimation and prediction. |
| Weight class of aircraft | Determination of wake-turbulence avoidance parameters. |
| Performance limitations | Determination of maneuver envelope. |
| Destination and waypoints | Flight-path planning. |
| Declared in-flight emergencies | Yield right of way to aircraft with emergency. |
| Severe weather or icing | Planning to avoid weather or to deal with icing. |
| Equipment failures | Accommodation of degraded mode of operation through choice of resolution option. |
| Detection of own aircraft by other aircraft | Determination of likelihood of other aircraft cooperating. |
| Additional proximate aircraft detected by other aircraft | Detection of multi-aircraft conflicts that may affect other aircrafts' options. |
| Relative data gathered by other aircraft | Comparison of relative position data to detect discrepancies. |
| Resolution plan which other aircraft is executing | Monitoring actions of other aircraft. |

The system design presented below could control a single aircraft within an OCC architecture. Since the main problem faced by the aircraft is conflict avoidance, most of the inference, communication, and planning deal with conflict recognition and resolution. The inputs to an aircraft are:

Own ship navigational information
  absolute position
  positions of all airports, navaids, and
    entry/exit fixes in the airspace
Localized situation displays
  onboard sensor derived information:
    x, y, z, heading, of all aircraft
    weather & terrain sensing
Structured messages from other aircraft.

The outputs from an aircraft are:

Commands to own aircraft.
Structured messages to other aircraft.

Each component of the design is given in a Pascal-like description followed by explanatory comments. We begin with the main, executive-control loop:

*Main Loop*

```
Main:
begin
     create initial plan;
     repeat  {look/plan/act loop}
        update-world-model (latest sensor information);
        expunge-request-list; {delete expired requests}
        repeat  {look for messages and resolve conflicts}
            if there are a new messages, process
             messages (message-list);
            if plan has not been checked against latest
             world update, then
            begin
                update a-priori plan for each goal;
                fix the plan;
            end;
            if there is a pending request and time to
             process it then process-messages
             (pending-request)
         until next update of "real world";
      until destination is reached;
end.
```

The main loop of our problem-solver is structured according to a basic problem-solving paradigm: sense → assess the situation → generate an initial plan → evaluate the plan → replan → act → monitor execution. In this case, however, the inner loop fixes plan conflicts and creates appropriate actions. If this were a centralized

problem-solver, this would not be a loop, but simply a procedure to fix conflicts and produce a set of actions based on the latest situation assessment. However, intermittent incoming messages may affect the plan—hence the iteration. After an initial check and resolution, the processor idles until a message arrives. A message triggers a new situation assessment, causing a new run through the look-revise-plan-create actions loop. When the loop quiesces, deferred actions, such as replying to low-priority requests, are processed. We shall now describe the sensor activities.

*Step 1: Update World Model*

```
Procedure update-world-model(new information);
begin
      parse the information, relating it to any prior requests;
      update the world model if possible;
      Compare observed actions of other aircraft with
        updated model of their intentions;
      if observations don't agree with your predictions,
      then
      begin
        try to infer intentions;
        if intentions cannot be deduced from
         present observations,
        then request intentions through communication;
        post a priority-labeled reminder;
      end;
      if a critical reply is still missing, then ask again with
        more urgency;
      if no reply at decision cut-off time, use trajectory
        projection;
end.
```

This is the first step of the main loop. Called with some new information, it attempts to associate the new information directly with its view of the world. Expected information items, such as tracking updates that match model projections, are simply incorporated in the model. Observations that do not agree with expectations are tagged and an attempt is made to resolve the discrepancy. Such resolution may involve either inferential reasoning or a request for data from the aircraft in question. Note that while most new information will be expected and will have little effect on the database, some information will spawn extensive inference. For example, information regarding an unseen aircraft causes the addition of that aircraft to the world model and a test for possible interactions between that aircraft and its plans or those of others.

*Step 2: Process Messages*

```
Procedure process-messages(message-list):
begin
      foreach message in message-list:
      begin
        parse the message;
```

```
        case
          info-request: if response is explicitly encoded
                or easily computed
                      then
                         if sufficient time exists
                         then reply with that
                         information else post
                         request for later reply;
          act-request:  post requested action in tentative
                plan;
          reply:  update-world-model(message);
          general info: update-world-model(message);
        end;
        delete message from message list;
      end;
end.
```

This loop handles all incoming message traffic, associating an action with each of the four types of messages: information requests, action requests, replies received from own requests, and general information. Messages that respond to requests generate updates in the world model. Requests for information either receive immediate replies or are posted for later reply. Requests for action are posted as constraints for the plan under construction. If these are low-priority constraints, they are the first to be eliminated during the replanning and conflict-resolution phase. Note that these items posted for later action always contain expiration information, either in the form of actual time (absolute or relative) or conditions that negate them.

*Step 3: Evaluate Plan*

```
Procedure evaluate-plan(plan);
begin
      while (create-ordered-list-of-conflicts) is not empty do
      begin
        resolve the highest-priority conflict;
        merge the resolution commands into the plan;
      end;
end.
```

This plan-correction routine embodies three important assumptions: (1) that conflicts vary in their importance, (2) that resolving one may resolve or create another, and (3) that merging disparate resolution suggestions may require an integration process.

The routine operates by looking for all conflicts, ranking them according to priority, solving the most important, then looking for any remaining conflicts. Frequently, solving the most important conflict will also solve the less important ones. However, solving the major conflict occasionally creates others, so we must allow for that possibility.

*Step 4: Create Ordered List of Conflicts*

```
Procedure create-ordered-list-of-conflicts;
begin
     repeat {find all conflicts}
       repeat {find a conflict}
          project ahead one time step using simulation
           that incorporates rules of "physics" with
           (incomplete) knowledge of the
           intentions of other aircraft;
        check for:
          separation violations;
          unfulfilled goals (e.g., too high when
           overflying the airport, or not
           heading in the right direction, or . . .);
        until found a conflict or looked far enough into the
          future;
      until looked far enough into the future;
      sort the list of conflicts according to rules of severity;
      communicate potential conflicts to other affected aircraft;
end.
```

This loop finds and ranks the potential airspace conflicts in terms of severity. Projecting ahead is done in a time-stepped fashion (an event-stepped alternative method is described below). Checking for conflicts can be done through a very large "case" or "switch" statement in which the alternative cases can be computed from conflict-determining predicates.

Sorting the list of conflicts implies some criterion function. How is severity determined? The simplest method relies on static orderings established a priori. In this case, the ordering can be based on a simple table lookup. However, more powerful techniques might use situation-dependent information to create dynamic rules of ordering. The utility of this more complex procedure depends on the situational complexity, dynamics, and solution sparseness in the task environment.

*Step 4a: Alternate Method for Creating Ordered List of Conflicts*

```
Procedure create-ordered-list-of-conflicts (alternate method);
begin
     repeat {find all conflicts}
          using analytic geometry techniques, compute
          expected time of next separation violation;
      until all conflicts before a certain time in future are detected;
      sort the list according to rules of severity;
      communicate potential conflicts to other affected aircraft;
end.
```

This alternative procedure may be used if the rules for determining a conflict are mathematically tractable (a situation that seldom occurs in the real world). In our ATC simulation, we may be able to predict easily and accurately the motions of aircraft because of the bounded, predictable domain.

This method is often more efficient than the time-stepped method. Routines can directly compute a list of "significant events" and their associated parameters—time of occurrence, aircraft involved, etc. Equations of motion can be used to update directly these interesting events, without incurring the costs of step-by-step updating and searching. This event-stepped form of simulation is favored if the cost of identifying and finding the "interesting events" is less than the cost of stepping through the corresponding time intervals.

*Step 5: Resolve a Conflict*

```
Procedure resolve-a-conflict;
begin
    case
    a "guaranteed" procedure for this conflict exists:
      return (that procedure);
    else
    begin
        foreach possible resolution command (from a table
         of applicable commands for this conflict type):
        begin
          Compute time interval of command issuances
           that would have prevented the conflict;
          Copy the simulated world and backdate to
           this span of time;
          Put the possible resolution commands in
           the plan;
          Fix the plan, evaluating the resulting
           look-ahead for n future time ticks;
          Save the results of that evaluation;
        end;
        return (best action according to the evaluation);
    end;
end.
```

Here we select one of the possible commands associated with the particular conflict type and try it out in the simulated world. Since we are assuming local, cooperative behavior, commands are typically limited to our own aircraft and possibly one other. We backdate the command as necessary to the most appropriate point of issuance. Trying the command out means calling the "fix plan" routine recursively with a new world and plan. Conflicts may occur with different frequencies, resulting in widely differing times for each branch of the search tree being generated. This process seems to be representative of the process humans use to search and evaluate in this domain. For example, in a tightly connected and complex conflict situation, human controllers look ahead 5 minutes or so, considering and resolving many potential conflicts. In a simpler situation, a controller may look much further ahead, with the same processing capabilities considering fewer conflicts.

Some of the commands considered will involve actions by other aircraft. The evaluations for these options will incur added costs of communication time delays and response uncertainty.

The result of this resolution function is a single "best" command for resolving the current conflict.

*Step 6: Merge the Resolutions*

```
Procedure merge-the-resolutions;
begin
      scan rules of command interaction, replacing
         the multiple commands with fewer that
         accomplish the same thing and are conflict-free;
      if another aircraft might help you reduce some of
         your non-goal-achieving actions
         then request such an action;
      if your intentions are significantly changed
         then notify neighbors;
      if your action conflicts with planned actions of other
         aircraft then bargain with the other aircraft;
end.
```

In the simplest case, the merge command merely links the set of resolution commands into an action sequence. The routine also notifies neighbors if any new actions have been added to the plan. More complex calculations are required if we consider replacement of the action sequence with an equivalent, smaller action set. Occasionally, an inefficient set of commands will be generated. The resolve routine finds plan modifications incrementally and considers each conflict separately. The merge routine may be able to look at the entire situation and prune the set of commands significantly.

Cooperating with other aircraft may also significantly improve the overall quality of solutions. An aircraft need only communicate those intentions that may affect others' behavior. Here a tradeoff between channel load and plan efficiency arises, some aspects of which are handled in the rules below.

*Step 7: Check Requested Action*

```
Procedure request-actions;
begin
      compute expected time delay associated with transmission
         and processing;
      if time delay is greater than decision time limit for
         action then delete request and replan actions;
      if channel occupancy is low then state full plan and
         request actions;
      if channel occupancy is high then request actions only;
end.
```

The routine simply checks to make sure the requested action can be performed within an allowable time span and without undue channel load. The expected time delay will depend on current channel occupancy, message length, and message-processing difficulty. The decision time limit is determined from the resolution routine.

*Step 8: Choose Between Inference and Communication*

```
Procedure infer-intentions: {observed behavior does not match
 model predictions}
begin
      search rulesets to determine if inference is possible
        for this situation;
      estimate time of processing;
      if sufficient time exists then activate inference
        process, associate confidence with result, and
        incorporate information in model;
      else request information from aircraft producing
        discrepancy;
end.
```

This is a simplified view of the inference/communicate decision. Inference is chosen if appropriate rulesets are present and time exists for the deductive process. Otherwise the information is requested from the other aircraft. Eventually, more sophisticated rules that consider relative time delays, confidences, and use of computing resources will be used to select between inference and communication in each instance.

*Step 9: Bargain with Other Aircraft*

```
Procedure bargain-with-other-aircraft;
begin
      compare plan(s) received from other aircraft with own
        plan (run each in look-ahead program and determine
        differences);
      if one plan dominates for all aircraft, adopt plan and
        acknowledge;
      if stalemate, attempt to find intermediate plan that
        reduces negative evaluation; if none available, use
        resolution procedure (highest overall evaluation, else
        use arbitrary rule);
end.
```

If plans by different aircraft are incompatible, the aircraft must compare the effects of the two (or more) plans, agree on a resolution procedure, and if a stalemate still exists, find a mutually satisfactory tie-breaking procedure.

# VI. IMPLICATIONS AND DIRECTIONS

We have examined several architectures for the distributed control of air fleets, and we have attempted to determine the environmental conditions that force architecture choices and influence system performance requirements. As a result of these investigations, we have postulated candidate architectures and distributed AI techniques for a number of different air fleet control environments.

Our six architectures, from the completely silent and individual object-centered autonomous to the multi-level hierarchical structure, represent variants of a single problem-solving "kernel." The system kernel appears to require the capabilities of sensing, inference, communication, world modeling, look-ahead, option generation, and execution, along with a means for scheduling these activities. The architectures differ in the physical form of the processes and the relative emphases placed on them. Certain architectures seem to rely primarily on inference and preplanned protocols, some focus on extensive communications, and some depend on exhaustive plan generation.

## SELECTION OF AN ARCHITECTURE

Our investigations suggest that at least four factors are critical in the selection of a distributed control architecture: (1) the degree of communication freedom, (2) the extent of vehicle clustering, (3) time stress, and (4) reliability requirements. The first of these, communication freedom, seems to be the dominating factor. If communications are highly overloaded, degraded strongly by distance or noise, or subject to countermeasures, the only alternative may be the object-centered autonomous architecture. This is the only architecture that can silently perform situation assessment, planning, and execution. If the range of constraints is relaxed somewhat, the object-centered cooperative architecture becomes suitable. This architecture minimizes communication distances by relying only on vehicle-to-vehicle transmissions within conflict groups. If the communications constraints are further relaxed, the local communications used in the space- and function-centered architectures are possible. Only when long-range communications are reliable and inexpensive can the global data fusion and planning processes of the plan-centered and hierarchical architectures be used.

The extent of vehicle clustering and time stress have secondary effects on choice of architecture. The aircraft to be coordinated may have natural lines of organization by geography, flight phase, goal, or level of abstraction. Selecting an architecture that corresponds to the form of clustering should maximize the ratio of fast, cheap computation to slow, costly inter-processor communication. Time stress reinforces the need for minimizing unnecessary communications. Also, the architecture implementations emphasizing rapid, local heuristic planning techniques (option pruning, abstraction, heuristic search) tend to be favored in time-stressed situations.

The presence of the fourth factor, reliability requirements, argues strongly for one of the object-centered architectures. These architectures are least affected by

loss of processors or communication links. In the cooperative version, messages can be routed along many different pathways. Also, the communication links tend to be close together, minimizing the possibilities of of jamming or detection in military situations.

## TRANSITIONS BETWEEN ARCHITECTURES

As aircraft progress through different phases of flight—takeoff, transition, enroute, and approach—it may be necessary to shift architectures. The aircraft may need to change from a loosely controlled object-centered architecture in the high-altitude enroute airspace to a highly structured space-centered architecture in the high-density terminal area. This transition may be accomplished through either (1) non-communicative, preset protocols, e.g., transition from OCC to space-centered at 6000 feet, (2) cooperative, local behavior, such as iteratively generating options, exchanging data, and bargaining about initiation points, or (3) designation of a higher-level decisionmaker to coordinate the transition. The transition itself may involve redistribution of data, modification of the communication networks, and/or shifts in the forms of planning and control.

The coordination of control hand-offs between sectors or regions of responsibility presents similar problems. Distribution of control introduces "seams" or boundary areas of overlapping or unspecified control responsibility. A large overlap of monitoring and control functions of neighboring processors will result in inefficient use of processing resources. Conversely, if the neighboring nodes exercise minimal coverage, some aircraft may "slip through the cracks." Techniques need to be developed for redundant processing at border areas, for synchronizing processing between regions, and for communicating information required for coordinating control hand-offs. These considerations are critical in all architectures in which a single node may control multiple objects.

## QUESTIONS FOR FUTURE RESEARCH

This work only begins to define the relationships among task environments, system architectures, and task performance. The questions that must still be addressed in further work include the following:

1. When should inference be used and when should communication be called on to update the world model? Can a single evaluation function for this choice adequately integrate time delays, data inaccuracies, reliability problems, and processing demands?
2. Can information-seeking and planning be supported by the same type of representation? How can information-seeking options that change the environment be represented?
3. What conflict-resolution procedures should be used when the goals of the different problem-solving nodes are in conflict? When should a separate node be designated as an arbitrator, and when should direct bargaining take place?

4. When should the focus of attention of a node be on search, conflict resolution, monitoring, etc.? What processes should be event-driven and what processes time-driven? How much scheduling responsibility should be given to the modules themselves?

## FUTURE RESEARCH

Several of the above questions will be addressed in a series of system implementations now in progress at Rand. We have modified an existing ATC terminal-area simulation (described briefly in Section V) to operate on Rand's PDP 11/45 and VAX computers. This simulation includes many of the decision and communication behaviors present in a high-density approach control sector. We are using the simulation to exercise groups of human controllers (to ascertain their heuristics for cooperative problem-solving) and to support demonstrations of automated distributed AI systems implemented on a DEC 2060 computer in INTERLISP.

We chose the OCC structure for much of our initial work, for the following reasons:

1. It is the most fine-grained of the candidate architectures, having only one controlled object per processor.
2. It supports the most interesting behavior (for example, autonomous, cooperative, and centralized planning are all possible).
3. It allows the greatest flexibility of physical representation: the processors may be onboard or remote, and different vehicles may assume different roles.
4. Higher levels of organization may easily be overlaid on the basic object-centered structure.

We are currently implementing a distributed set of kernel OCC nodes in the simulation. Each of these automated nodes will control an aircraft and will interact to produce coordinated behavior. We are implementing rulesets for:

1. *Model updating*—incorporating new information in the world model.
2. *Inference*—determining intentions of other aircraft through deduction.
3. *Initial plan generation*—synthesizing candidate plans.
4. *Option evaluation*—using look-ahead to evaluate options.
5. *Communications management*—opening channels and formatting messages.
6. *Replanning*—using rulesets to generate responses to potential conflicts.
7. *Bargaining*—using protocols to determine precedence of aircraft.
8. *Scheduling of activities*—transitioning among rulesets 1 through 8.

Subsequent efforts will focus on expansion of this baseline distributed system to deal with communication range limitations, noise, and environmental uncertainty. This work is essential to the identification of problems of physical realization—processor cycle time, communications range limitations, memory constraints, etc.—specific to each architecture. Also, we are initiating parallel efforts with human controllers acting in place of computer-based problem-solvers to help elucidate additional features of the various architectures.

So far we have concentrated on developing entirely automated distributed systems. Most real-world systems will be hybrid, making use of the special capabilities of both humans and machines. We must integrate our work with models of the human cognitive processes involved in cooperative behavior and with developments in interactive display techniques and knowledge elicitation procedures. We also need to investigate the problems of coordinating many independent, asynchronous processes, where the processes may be human-initiated or machine-initiated. Our findings should directly affect engineering applications and stimulate new areas of technology in distributed intelligent systems.

# Appendix

# CALCULATIONS OF REDUCTION OF CONFLICT TESTS BETWEEN GLOBAL AND LOCAL PROBLEM RECOGNITION

In this appendix, we calculate the difference between the number of conflict tests required in global problem recognition and the number in local problem recognition. Both cases require monitoring several tolerance ranges, including aircraft separation, terrain clearance, and equipment performance. Global problem recognition requires tests for possible conflicts between all objects, while local problem recognition considers conflicts only within clusters of objects.

Global problem recognition may be performed for an airspace by collecting state data from all sensor nodes and checking tolerances with respect to all objects and all constraints. Looking at aircraft-to-aircraft separation checks alone, the number of individual tests on n objects is

$$\binom{n}{2}$$

The possibility of multi-party conflicts of three or more aircraft rapidly increases the number of tests. Because of stricter separation criteria resulting from a decrease in maneuvering space (Andrews and Hollister, 1980), such conflicts may need to be treated differently from conflicts of two aircraft. The maximum number of conflict tests increases to

$$\sum_{r=2}^{n} \binom{n}{r} = 2^n - (n+1)$$

With large n, the number of conflict tests then approximates $2^n$. We expect this exhaustive, global problem-recognition procedure to be necessary (1) if the aircraft separation requirements are large compared to the size of the airspace and (2) if replanning actions normally involve most of the aircraft in the airspace.

In local problem recognition, only interactions within a cluster of neighboring aircraft must be considered, so the number of interactions is reduced considerably. For example, suppose the aircraft are clustered into m loosely coupled conflict groups with n(k) elements in the $k^{th}$ group. The number of two-body conflict checks required is

$$\sum_{k=1}^{m} \binom{n(k)}{2}$$

If higher-order interactions are possible, the number of potential conflicts becomes

$$\sum_{k=1}^{m} \sum_{r=2}^{n(k)} \binom{n(k)}{r} = \sum_{k=1}^{m} 2^{n(k)} - (n+m)$$

With large n, the number of conflict tests then tends to

$$\sum_{k=1}^{m} 2^{n(k)}$$

This is much smaller than the $2^n$ tests required for global problem recognition. The potential savings realized through clustering should increase further if boundary, terrain, fuel, or weather constraints are included in the problem-recognition task.

Formation of the clusters is basically a problem of minimizing dependencies between groups. We can perhaps most easily define the clusters by grouping the aircraft according to physical proximity. The procedure for this clustering is an adaptation of the nearest-neighbor algorithm used in pattern classification. We first define a distance metric based on some linear combination of horizontal separation and altitude. We specify a program to compute the distance between every pair of aircraft in the airspace.* The program determines the shortest distance between two aircraft, puts the aircraft so joined into a single cluster, then determines the next shortest distance, and so on. We consider each cluster to be a conflict group. Using trajectory projection, the method currently used in ATC conflict-prediction programs, we can test all aircraft within each group for possible conflicts. Potential conflicts across groups are not checked.

In a similar fashion, the number of options searched by a distributed planning node may be considerably smaller than the number searched by a centralized node. The following simple example illustrates this.

A controller must maintain clearance between 12 aircraft in a sector. For simplicity, let us assume that he may choose only five altitudes and five headings. In addition, he may activate these commands at any of four time steps. At any time, the number of options is

$$\text{Number of options} = (\text{N aircraft}) \times (\text{K headings} + \text{L altitudes}) \times (\text{M initiation times})$$

Under the conditions of this example, this represents an exhaustive search of 480 options, each one of which consists of a single action by a single aircraft. If a conflict situation requires three sequenced actions for resolution, as many as $(480)^3$ option sequences need to be tested. Reduction of the problem into four independent conflict clusters of three aircraft simplifies the problem considerably, even if each cluster must still perform three sequenced actions. Each cluster then checks a maximum of $(120)^3$ option sequences, a reduction by a factor of 64 over the global procedure. Further reductions with distribution are expected from truncation of the time horizon because of faster local replanning and from assigning different portions of the planning to each of the several aircraft in a cluster.

*Actually, more efficient means are possible for this computation (see Uhr, 1973).

# GLOSSARY OF ATC TERMS

**approach control.** The ATC facility responsible for radar separation and coordination of aircraft in the vicinity of an airport. Approach control's typical jurisdiction extends about 20 to 40 miles from the airport horizontally and 5 to 10 miles vertically.

**cluster.** A group of aircraft considered as a single interacting body. The clustering dimensions may include distance, aircraft type, and flight phase.

**conflict.** A situation in which one or more aircraft violate constraints of altitude and horizontal separation.

**enroute control.** The ATC facility responsible for aircraft separation and coordination while in cruising flight between departure and destination airports.

**fix.** A specified geographic location used in navigation. Usually, fixes are easy-to-locate points formed by radio beacons.

**hand-off.** Release of control of an aircraft, giving it to another ATC facility as the aircraft proceeds past the airspace limits of the control jurisdiction.

**link.** A data communication between aircraft or between processing nodes.

**node.** An individual processor which may perform sensing, situation assessment, planning, or execution. The node may be onboard an aircraft or may be ground-based.

**strategic planning.** Long-time-horizon planning dealing with both local and distant aircraft interactions.

# BIBLIOGRAPHY


Andrews, J., and W. Hollister, *Electronic Flight Rules: An Alternative Separation Assurance Concept*, Lincoln Laboratory Report No. FAA-RD-80-2, Lexington, Massachusetts, January 1980.

Babich, A., J. Grason, and D. Parnas, "Significant Event Simulation," *Communications of the ACM*, Vol. 18, No.6, 1975, pp. 323-329.

Brooks, R., and V. Lesser, *Distributed Problem Solving Using Iterative Refinement*, COINS Technical Report 79-14, University of Massachussetts, Amherst, May 1979.

Clark, D., K. Pogram, and D. Reed, "An Introduction to Local Area Networks," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1539-1548.

Cohen, D., J. Barnett, Y. Yemini, and D. Schwabe, *DSN—Distributed Sensor Networks*, Information Sciences Institute, ISI-WP-12, Marina del Rey, California, April 1979.

Couluris, G., M. Tashker, and M. Penick, *Policy Impacts of ATC Automation: Human Factors Considerations*, SRI International and Payne-Maxie Consultants, Report No. FAA-AVP-78-1, January 1978.

Dalkey, N., *Group Decision Theory*, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7749, July 1977.

Erman, L., F. Hayes-Roth, V. Lesser, and D. Reddy, "The Hearsay-II Speech Understanding System: Integrating Knowledge to Reduce Uncertainty," *Computing Surveys*, June 1980, pp. 213-252.

Farris, D., and A. Sage, "Introduction and Survey of Group Decision Making with Applications to Worth Assessment," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-5, No. 3, May 1975, pp. 346-358.

Hayes-Roth, B., and F. Hayes-Roth, *Cognitive Processes in Planning*, The Rand Corporation, R-2366-ONR, December 1978.

Hayes-Roth, F., "The Role of Partial and Best Matches in Knowledge Systems," in D. A. Waterman and Frederick Hayes-Roth (eds.), *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

Hummel, R., and S. Zucker, *On the Foundations of Relaxation Labeling Processes*, Computer Vision and Graphics Laboratory, McGill University, TR-80-7, July 1980.

Kahn, R., S. Gronemeyer, J. Burchfiel, and R. Kunzelman, "Advances in Packet Radio Technology," *Proceedings of the IEEE*, Vol. 66, No. 11, November 1978, pp. 1468-1496.

Kelley, J., *Distributed Processing Techniques for En Route Air Traffic Control*, The Mitre Corporation, TR-7589, July 1977.

Kinney, G., M. Spahn, and R. Amato, *The Human Element in Air Traffic Control: Observations and Analyses of the Performance of Controllers and Supervisors in Providing ATC Separation Services*, The Mitre Corporation, Technical Report MTR-7655, December 1977.

Kreifeldt, J., L. Parkin, P. Rothschild, and T. Wempe, "Implications of a Mixture of Aircraft With and Without Traffic Situation Displays for Air Traffic Management," presented at the Twelfth Annual Conference on Manual Control,

University of Illinois, Urbana, Illinois, and Ames Research Center, Moffett Field, California, May 1976.

Lesser, V., and L. Erman, "Distributed Interpretation: A Model and Experiment," *IEEE Transactions on Computers*, Vol. C-29, No. 12, December 1980, pp. 1144-1162.

Marschak, J., "The Payoff-Relevant Description of States and Acts," *Journal of Econometrics*, Vol. 21, No. 4, October 1973, pp. 719-725.

Martin, J., *Developments in Tele-Communications*, Prentice-Hall, New Jersey, 1977.

McClellan, J., "Understanding the Future ATC System," *Business and Commercial Aviation*, September 1978.

Mills, R., N. Aume, and R. Bachert, *Summary Report of AMRL Remotely Piloted Vehicle (RPV) System Simulation*, Aerospace Medical Research Laboratory, Wright-Patterson AFB, AMRL-TR-75-126, December 1975.

Neubauer, J., "U.S. Cruise Missile Development," *Astronautics and Aeronautics*, Vol. 17, No. 9, September 1979, pp. 24-35.

Newell, A., and H. Simon, *Human Problem-Solving*, Prentice-Hall, New Jersey, 1972.

Nilsson, N., *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

Parnas, D., "On a 'Buzzword': Hierarchical Structure," in J. L. Rosenfeld (ed.), *Information Processing 74*, North-Holland Publishing Company, 1974, pp. 336-339.

Phatak, A., N. Gupta, and I. Segall, *Analysis of Controller/System Dynamics for a Remotely Piloted Vehicle Strike Mission*, Systems Control, Inc., AMRL-TR-74-80, September 1974.

Ratcliffe, S., "Principles of Air Traffic Control," in A. Benoit (ed.), *A Survey of Modern Air Traffic Control*, NATO AGARDograph, No. 209, Vol. 1, July 1975, pp. 3-20.

Rucker, R., *Automated Enroute ATC (AERA): Operational Concepts, Package 1 Description, and Issues*, The Mitre Corporation, MTR-79W00167, May 1979.

Sacerdoti, E., "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence*, Vol. 5, No. 2, 1974, pp. 115-135.

Smith, R., *A Framework for Problem Solving in a Distributed Processing Environment*, Stanford University, Computer Science Department Report No. STAN-CS-78-700, December 1978.

Smith, R., and R. Davis, "Distributed Problem Solving: The Contract Net Approach," *Proceedings of the Second National Conference of the Canadian Society for Computational Studies of Intelligence*, Toronto, July 1978, pp. 278-287.

Stefik, M., *Planning with Constraints*, Computer Science Department, Stanford University, Doctoral Dissertation, January 1980.

Uhr, L., *Pattern Recognition, Learning, and Thought*, Prentice-Hall, New Jersey, 1973.

Waterman, D. A., and Frederick Hayes-Roth (eds.), *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

Wesson, R., *Problem-Solving with Simulation in the World of an Air Traffic Controller*, University of Texas at Austin, Doctoral Dissertation, 1977.

Wesson, R., and F. Hayes-Roth, *Dynamic Planning: Searching Through Time and Space*, The Rand Corporation, P-6266, February 1979.

Wesson, R., and F. Hayes-Roth, with J. Burge, C. Stasz, and C. Sunshine, *Network Structures for Distributed Situation Assessment*, The Rand Corporation, R-2560-ARPA, August 1980.

[redacted]